



## **SISTEM INTELIGENT PENTRU MONITORIZAREA ȘI CREȘTEREA PLANTELOR ÎN GHIVECI**

LUCRARE DE LICENȚĂ

Absolvent: **Iulia – Maria CORNEA**

Coordonator **Asis. Dr. Ing. Cosmina – Daniela IVAN**  
științific:

**2020**



FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE  
DEPARTAMENTUL CALCULATOARE

DECAN,  
**Prof. dr. ing. Liviu MICLEA**

DIRECTOR DEPARTAMENT,  
**Prof. dr. ing. Rodica POTOLEA**

Absolvent: **Iulia – Maria CORNEA**

**TITLUL LUCRĂRII DE LICENȚĂ**

1. **Enunțul temei:** *Sistem inteligent ce combină tehnologia IOT și Web pentru monitorizarea și susținerea creșterii unei plante în ghiveci.*
2. **Conținutul lucrării:** *Pagina de prezentare, Introducere-Contextul proiectului, Obiectivele Proiectului, Studiu Bibliografic, Analiză și fundamentare teoretică, Proiectare de Detaliu și Implementare, Testare și Validare, Manual de Instalare și Utilizare, Concluzii, Bibliografie.*
3. **Locul documentării:** Universitatea Tehnică din Cluj-Napoca, Departamentul Calculatoare
4. **Consultanți:**
5. **Data emiterii temei:** 1 noiembrie 2019
6. **Data predării:** 10 septembrie 2020

Absolvent: \_\_\_\_\_

Coordonator științific: \_\_\_\_\_



FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE  
DEPARTAMENTUL CALCULATOARE

**Declarație pe proprie răspundere privind  
autenticitatea lucrării de licență**

Subsemnata **Cornea Iulia Maria**, legitimat(ă) cu CI seria CJ nr. 346184 CNP 2930525125824, autorul lucrării **Sistem intelligent pentru monitorizarea și creșterea plantelor în ghiveci** elaborată în vederea susținerii examenului de finalizare a studiilor de licență la Facultatea de Automatică și Calculatoare, Specializarea Calculatoare din cadrul Universității Tehnice din Cluj-Napoca, sesiunea septembrie a anului universitar 2019 - 2020, declar pe proprie răspundere, că această lucrare este rezultatul propriei activități intelectuale, pe baza cercetărilor mele și pe baza informațiilor obținute din surse care au fost citate, în textul lucrării, și în bibliografie.

Declar, că această lucrare nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române și a convențiilor internaționale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în fața unei alte comisii de examen de licență.

In cazul constatării ulterioare a unor declarații false, voi suporta sancțiunile administrative, respectiv, *anularea examenului de licență*.

Data

10.09.2020

Nume, Prenume

Cornea, Iulia – Maria

Semnătura

---

## Cuprins

<b>Capitolul 1. Introducere.....</b>	<b>1</b>
1.1. Contextul proiectului .....	2
<b>Capitolul 2. Obiectivele Proiectului .....</b>	<b>5</b>
2.1. Cerințe funcționale.....	6
2.1.1. Autentificarea .....	6
2.1.2. Navigarea prin lista plantelor stocate în sistem .....	6
2.1.3. Crearea unei grădini virtuale .....	6
2.1.4. Adăugare de plante noi în grădina virtuală.....	7
2.1.5. Vizualizarea nevoilor unei plante .....	7
2.1.6. Afisarea datelor despre plantele din grădina virtuală .....	7
2.1.7. Monitorizarea mediului .....	7
2.1.8. Păstrarea umidității solului adekvată .....	8
2.2. Cerințe non-funcționale .....	8
<b>Capitolul 3. Studiu Bibliografic.....</b>	<b>9</b>
3.1. Proiectarea hardware .....	9
3.1.1. Placa de dezvoltare .....	10
3.1.2. Dispozitive periferice .....	13
3.2. Proiectare software .....	14
<b>Capitolul 4. Analiză și Fundamentare Teoretică.....</b>	<b>16</b>
4.1. Cazuri de utilizare.....	18
4.1.1. Înregistrare.....	19
4.1.2. Autentificare .....	20
4.1.3. Vizualizare listă globală de plante .....	21
4.1.4. Adăugare plantă în lista globală .....	21
4.1.5. Vizualizare grădina virtuală.....	22
4.1.6. Adăugare plantă în grădina virtuală.....	22
4.1.7. Creare tip de plantă local .....	23
4.1.8. Vizualizare infomării plantă din grădină .....	24
4.1.9. Proces de achiziție a Grădinarului de ghiveci (modulul hardware).....	25
4.1.10. Vizualizare date despre mediul planșei .....	26
4.2. Module.....	28
4.2.1. Modulul de gestionare a utilizatorilor .....	28

---

4.2.2. Modulul de gestionare a plantelor .....	28
4.2.3. Modulul de grădinărit semi-automat .....	29
<b>Capitolul 5. Proiectare de Detaliu si Implementare.....</b>	<b>32</b>
5.1. Aplicația server.....	33
5.1.1. Kotlin .....	35
5.1.2. Gradle .....	36
5.1.3. Spring boot .....	36
5.1.4. OpenAPI .....	37
5.1.5. Sendgrid.....	40
5.1.6. Spring Data JPA .....	40
5.2. Baza de date .....	40
5.2.1. PostgreSQL.....	40
5.2.2. Flyway .....	41
5.3. Aplicația client.....	42
5.3.1. Componentele aplicației .....	43
5.3.2. OpenAPI, npm, Verdaccio.....	44
5.4. Modulul Hardware.....	45
5.4.1. Submodulul de grădinărit – ATmega382P .....	47
5.4.2. Submodulul de comunicare – ESP8266 .....	50
5.4.3. Comunicare dintre submodulele modulului hardware.....	51
5.5. Alte instrumente.....	51
5.5.1. Git .....	52
5.5.2. IntelliJ IDEA.....	52
5.5.3. Arduino Software (IDE) .....	52
5.5.4. Heroku .....	52
<b>Capitolul 6. Testare și Validare.....</b>	<b>53</b>
6.1. Testarea manuală .....	53
6.2. Testarea automată .....	55
6.3. Testarea componentelor.....	56
6.3.1. Testarea plăcuței R3 UNO + WiFi .....	56
6.3.2. Testarea senzorului DHT11 .....	61
6.3.3. Testarea modulului de măsurarea a umidității solului.....	62
6.3.4. Testarea fotorezistorului .....	63
6.3.5. Testare mini pompa de apă .....	64

---

<b>Capitolul 7. Manual de Instalare si Utilizare .....</b>	<b>66</b>
7.1. Resurse necesare pentru instalare.....	66
7.2. Manual de utilizare pentru utilizator neautentificat.....	66
7.3. Manual de utilizare pentru utilizator înregistrat .....	69
7.4. Manual de utilizare pentru administrator.....	71
7.5. Manual de utilizare pentru utilizator hardware.....	73
<b>Capitolul 8. Concluzii .....</b>	<b>76</b>
8.1. Contribuții personale .....	76
8.2. Analiza rezultatelor obținute.....	76
8.3. Dezvoltări ulterioare .....	77
8.3.1. Dezvoltări software.....	77
8.3.2. Dezvoltări hardware .....	77
<b>Bibliografie .....</b>	<b>79</b>
<b>Anexă 1 – Lista figurilor .....</b>	<b>81</b>
<b>Anexă 2 – Lista tabelelor .....</b>	<b>84</b>
<b>Anexă 3 – Glosar de termeni.....</b>	<b>85</b>

## Capitolul 1. Introducere

Acum aproximativ 7.000 - 10.000 de ani, undeva în Semiluna fertilă (zonă din vestul Asiei, parte care include Mesopotamia și Levantul) homo sapiens a început agricultura de subzistență. După cum prezintă Harari în „Sapiens”[1], dacă până atunci oamenii erau obișnuiți să își ia sursa de hrana din ce voia natura să le ofere, acum, omul începe să controleze natura astfel încât aceasta să îi ofere ce vrea el. Acum, omul a început să își pregătească pământul, să semene grâul, să îl crească, îngrijească, să îl ferească de prădători, să îl recolteze, depoziteze, ca mai apoi să îl consume.

După 10.000 de ani, agricultura, întregul proces de creștere a plantelor, este de nerecunoscut. Deși în esență e vorba de parcurgerea acelorași etape, modul și viteza cu care acestea sunt parcuse în ziua de azi este cu totul diferit. În ultimele sute de ani agricultura a evoluat substanțial. Cea mai de ne recunoscut schimbare fiind automatizarea acesteia. Serele folosesc sisteme automate de monitorizare și calibrare a temperaturii, umidității din aer, umidității solului, a lumini, etc pentru a se asigura că obțin cea recolta cea mai bună în timpul cel mai scurt și cu cele mai mici costuri.

Ingineri și botaniști specializați proiectează sere ce caută în continuă dobândirea celei mai ample recolte cu cel mai redus buget. Unele dintre cele mai ostentative alterări aduse procesului de creștere a plantelor sunt verticalizarea culturilor, înlocuirea solului cu apă și folosirea luminii artificiale de creștere. Acestea sunt și modificările care aduc cel mai mult profit. Prin agricultura pe verticală, aceeași bucată de pământ poate produce de 4 – 5 sau chiar 10 ori mai mult. Înlocuirea solului cu apă face posibilă dezvoltarea recoltei indiferent de zona geografică și resursele existente. Plasarea de lumini artificiale (la distanțe precis calculate de fiecare plantă) asigură dezvoltarea plantelor în cel mai rapid mod. Alte modificări aduse procesului de cultivare a plantelor sunt controlul temperaturii aerului sau controlul umidității aerului sau al solului. Acestea sunt posibile datorită segmentului de IOT (Internet of Things).

Noțiunea de IOT a evoluat și și-a schimbat definiția de-a lungul anilor dar în ultimii ani am putut să ajungem cel puțin la consensul că Internet of Things este un sistem care interconectează dispozitive de calcul cu mașinării mecanice și/sau digitale identificabile în mod unic, care are abilitatea să transfere informație în cadrul rețelei fără a necesita interacțiuni între oameni sau între om și calculator[2].

La scară largă, pentru practicarea agriculturii, am văzut că există soluții tehnice și proiectanți specializați care pot implementa aceste soluții.

La scară mică, putem vorbi de agricultura de subzistență iar la scară micro, de creșterea plantelor pentru decor. În ambele cazuri putem spune că în majoritatea timpului nu avem proiectanți specializați și nici de tehnologii IOT care să optimizeze procesul de creșterea a plantelor. Pentru prima lipsă, cea a personalului specializat, nu putem veni cu o soluție ușoară, dar pentru cea de-a doua, soluțiile încep să apară la prețuri tot mai accesibile. În continuare ne vom axa pe creșterea plantelor pentru decor, mai exact, pe creștea plantelor de ghiveci și cum poate tehnologia IOT să facă această slujbă mai ușoară și mai sigură.

## 1.1. Contextul proiectului

Creșterea plantelor pentru decor, în special a platelor de ghiveci în apartamente este influențată de mai mulți factori, câțiva dintre aceștia meritând a fi amintiți cum ar fi luminozitatea de care beneficiază planta, tipul solului în care aceasta este plantată, dimensiunea ghiveciului comparativ cu dimensiunea rădăcinilor și cel mai important, udarea corespunzătoare. În timp ce primii trei factori pot fi corectați într-o acțiune singulară, asigurarea cantității optime de apă pentru plante necesită un efort constant. Udarea deficitară a acestora este cauza principală pentru care plantele de apartament se ofilesc și mor.

În salvarea vegetație de apartament vin sistemele inteligente din lumea IOT care oferă o varietate de soluții, începând de la notificări pentru utilizator ca acesta să își ude plantele, până la automatizarea completă a procesului de creștere a plantelor.

Un sistem IOT este format atât din componente hardware cât și software. Pentru a oferi o oferă o imagine de ansamblu asupra lumii IOT, atât am ales să o împart în patru subcategorii ca apoi să evidențiez rolul și compoziția fiecarei. După cum se observă și în

Figura 1.1 categoriile sunt: Dispozitivele de măsurare, Dispozitivele acționabile, Logica de control (împreună cu dispozitivele de stocare a acesteia) și Interfața cu utilizatorul (pentru care sunt necesare și dispozitive cu care utilizatorul se poate conecta la respectiva interfață).

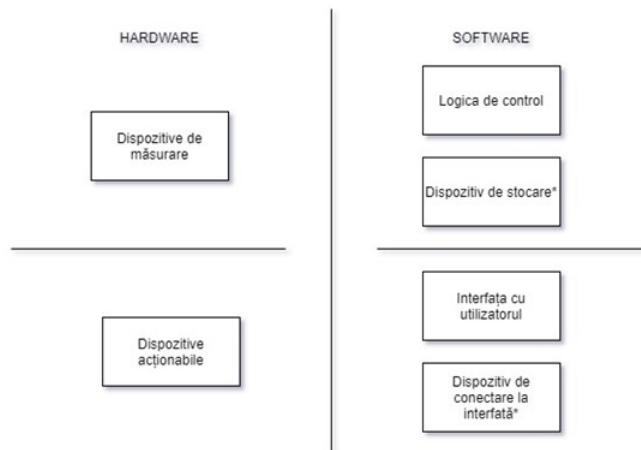


Figura 1.1 Componete ale unui sistem IOT

Prima componentă hardware sunt dispozitivele de măsurare și ele se referă la orice dispozitiv ce poate lua informație din mediu și o poate converti în informație analogică sau digitală ce poate fi interpretată de logica de control. Exemple populare de dispozitive de măsurare ar fi senzorii sau camerele de luat imagini. În majoritatea cazurilor aceste dispozitive sunt capabile să culeagă și transmită informația mai departe în regim autonom, fără interacțiunea unui utilizator. Aceasta este și trăsătura principală a unui sistem IOT, faptul că poate funcționa în mod independent de interacțiune umană, de aceasta fiind nevoie doar pentru instalare și calibrare.

A doua componentă hardware e reprezentată de dispozitivele acționabile. În cadrul unui sistem IOT, ele sunt acele dispozitive care sunt controlate de sistem și care pot aduce modificări mediului. Exemple reprezentative ar fi sistemele de iluminat, pompele de apă, generatoarele de căldură, etc. Acestea sunt acționate de logica de control.

Logica de control dintr-un sistem IOT este partea software care se poate ocupa cu recepționarea informației de la dispozitivele de măsurare, recepționarea comenzi de la utilizator, procesarea acestor date și/sau comenzi și controlul dispozitivelor acționabile în funcție de rezultatul acestei procesări. Logica de control poate fi centralizată pentru întreg sistemul și stocată în cloud sau pe un server în rețea, dar ea poate fi de asemenea și decentralizată, fiecare dispozitiv venind cu propria logică de control. În exemplul din Figura 1.2 observăm o seră intelligentă în care logica de control e stocată în cloud, agregă date de la mai mulți senzori – „Wireless Environmental Sensors” – și controlează mai multe dispozitive – „Fans”, „Smart Hydroponics”, „CO<sub>2</sub> Generator”, „Air Conditioning” – ce pot altera starea mediului. În acest caz nu e nevoie de interacțiunea umană.

## IoT Smart Greenhouse

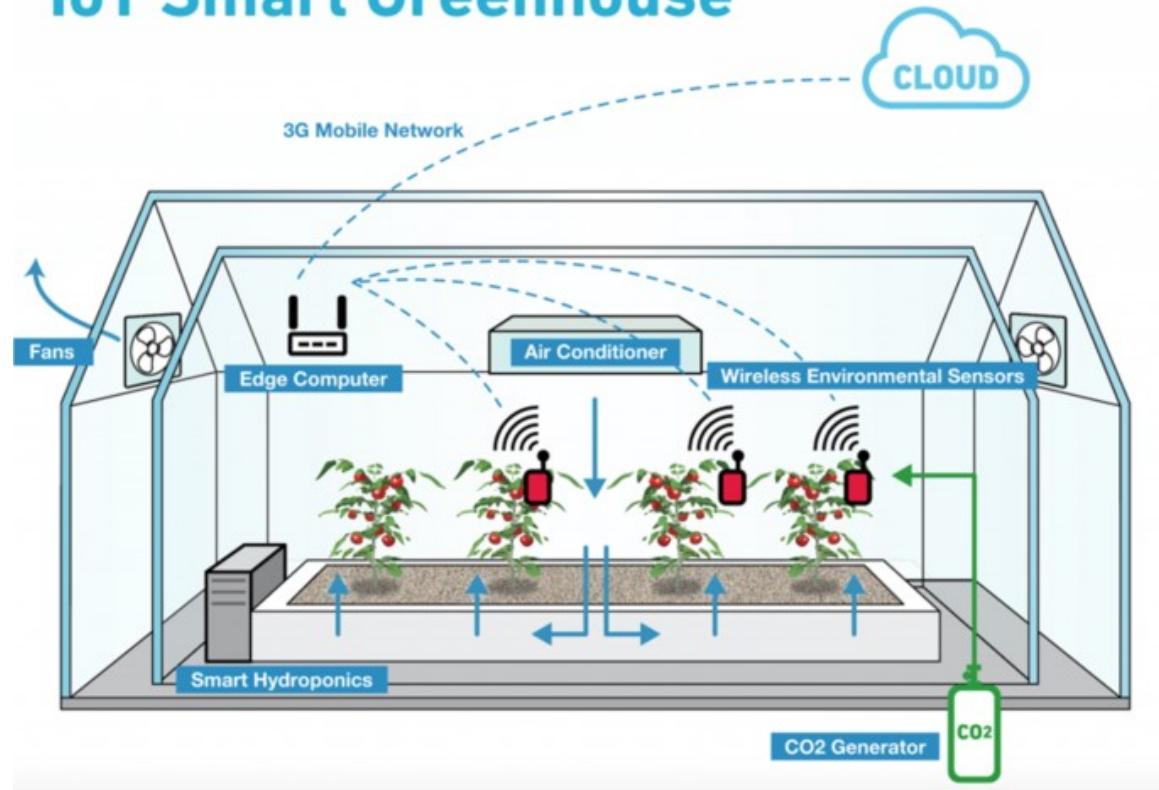


Figura 1.2 Seră intelligentă<sup>1</sup>

Spre deosebire de Figura 1.2 în Figura 1.3 avem un concept de sistem IOT în care logica de control e stocată direct pe dispozitivul IOT iar interacțiunea utilizatorului e

<sup>1</sup> <https://aws.amazon.com/blogs/iot/aws-iot-driven-precision-agriculture/>

necesară pentru ca acest sistem IOT să aibe sens. Mai precis, în acest caz avem un ceainic inteligent ce dispune de senzor de temperatură a apei și care poate fi controlat de la distanță prin intermediul unui interfețe accesibilă cu un smartphone. Utilizatorul poate modifica mediul (temperatura apei din ceainic) prin plasarea unei comenzi în cadrul interfeței, care la rândul ei va comanda dispozitivul acționabil – ceainicul – să încălzească apa până la o anumită temperatură.



Figura 1.3 Sistem IOT cu ceainic intelligent<sup>2</sup>

Ultima subcategorie pe care o prezentăm în cadrul unu sistem IOT este interfața pentru utilizator. Aceasta e deseori reprezentată de o aplicație web sau de o aplicație mobile care are ca scop facilitarea interacțiuni dintre utilizator și sistemul inteligent. Funcțiile pe care interfața le oferă sunt, de regulă, funcții de monitorizare și opțional de controlare a mediului sistemului. În exemplele noastre funcțiile de monitorizare ar putea oferi informații cu privire la nivelul de CO<sub>2</sub> din seră sau la temperatura apei din ceainic, în timp ce funcțiile de alterare a mediului ar putea modifica nivelul de CO<sub>2</sub> din seră sau temperatura apei din ceainic.

---

<sup>2</sup> <https://depositphotos.com/184063154/stock-illustration-hand-controlling-kettle-with-smartphone>

## Capitolul 2. Obiectivele Proiectului

Lucrarea de față își propune proiectarea unui sistem intelligent de grădinărit de tipul IOT pentru monitorizarea și automatizarea parțială a creșterii plantelor de apartament. În Figura 2.1 este ilustrat succint compoziția unui astfel de sistem de grădinărit. El se compune din:

- Componente hardware de măsurare și alterare a mediului.
- Aplicație de tip server pentru stocare de date
- Aplicație de tip client pentru interacțiunea utilizatorului cu sistemul

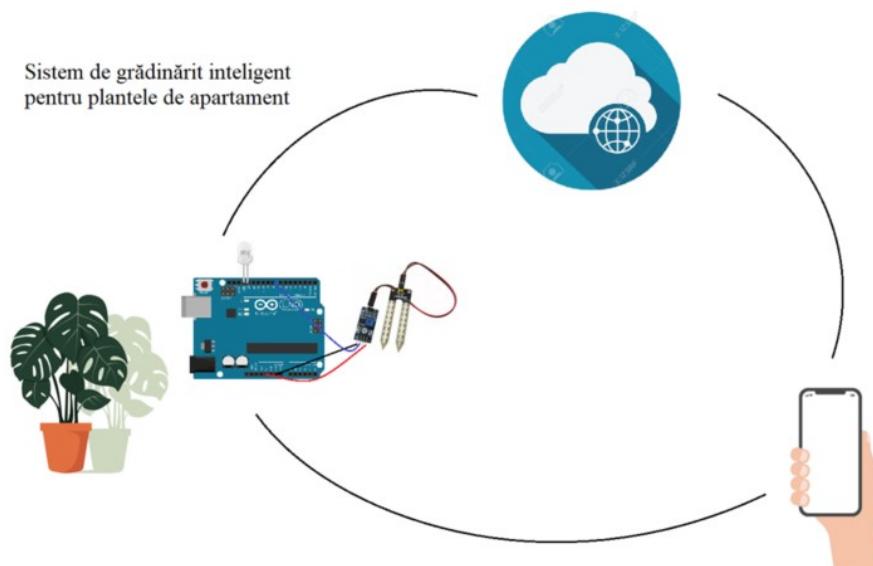


Figura 2.1 Sistem de grădinărit inteligent

Scopul principal ce se dorește atins este acela de a oferi un prototip care poate să crească diverse plante în ghiveci cu interacțiune minimă din partea utilizatorului. Scopul e de a crește plantele dintr-un singur ghiveci cu un sistem de grădinărit și de a monitoriza creșterea acestora. Scopul secundar este a pune la dispoziție utilizatorilor care nu au acces la partea de hardware a sistemului de grădinărit intelligent o aplicație care facilitiază creșterea plantelor.

Lucrarea de față va încerca să își atingă obiectivele prin următoarele:

- Monitorizarea mediului
- Controlul mediului
- Stocarea și deservirea de informații despre plante și mediul acestora
- Oferirea unei interfețe web prin care utilizatorul poate comunica cu sistemul

## 2.1. Cerințe funcționale

Pentru a-și atinge obiectivele, sistemul oferă funcționalități care implică interacțiunea utilizatorului pe tot procesul scenariului, cât și funcționalități care implică interacțiunea limitată a utilizatorului cu sistemul pe parcursul unui scenariu. Funcționalitățile care implică interacțiunea utilizatorului pe tot parcursul scenariului de utilizare sunt acelea care au ca scop stocarea de informații din partea utilizatorului și deservirea acestuia cu informații. Ele sunt:

- Autentificarea
- Navigarea prin lista plantelor stocate în sistem
- Crearea unei grădini virtuale cu plantele personale ale utilizatorului
- Adăugarea de plante noi care nu sunt în sistem în grădina virtuală
- Vizualizarea nevoilor unei plante (stocate în sistem sau create de utilizator)
- Afisarea datelor despre mediul plantelor din grădină

Scenariile ce implică o interacțiune limitată a utilizatorului cu sistemul au în vedere creșterea eficientă a plantelor. Acestea sunt scenariile care necesită partea de hardware a sistemului de grădinărit. Ele sunt:

- Monitorizarea mediului
- Păstrarea umidității soului adecvată

### 2.1.1. Autentificarea

Funcționalitatea de autentificare permite utilizatorului să se logheze în cadrul sistemului și să salveze date despre plantele sale. Aceste date sunt persistante și sunt accesibile doar utilizatorului în cauză.

### 2.1.2. Navigarea prin lista plantelor stocate în sistem

Sistemul beneficiază de o bază de date cu cele mai întâlnite plante de apartament din zona geografică a României. Utilizatorii pot naviga prin această listă de plante pentru a identifica planta pe care vor să o crească cu ajutorul sistemului de grădinărit înțeligențial sau cu scopul de a-și alege o plantă în funcție de nevoile acesteia și mediul pe care îl poate oferi.

### 2.1.3. Crearea unei grădini virtuale

Pentru a ține o mai bună evidență a plantelor din apartament, utilizatorul își poate crea o grădină virtuală în care să stocheze plantele pe care le deține. De notat că pentru a monitoriza mai multe ghivece, e nevoie de mai multe prototipuri de sisteme inteligente de grădinărit (un sistem de grădinărit, întreține doar plantele dintr-un singur ghiveci). Această funcționalitate de creare a unei grădini virtuale oferă posibilitatea utilizatorului să țină o evidență a frecvenței de udare și a plantelor care nu beneficiază de un sistem de grădinărit înțeligențial.

#### *2.1.4. Adăugare de plante noi în grădina virtuală*

Întrucât sistemul nu deține toate plantele existente, această opțiune de a adăuga alte plante oferă utilizatorului posibilitatea să stocheze în grădina sa virtuală o plantă pe care să o descrie manual. A descrie manual o plantă înseamnă în acest caz descrierea nevoilor plantei respective. Nevoile de care ține cont sistemul sunt: temperatura aerului, umiditatea aerului, nivelul de luminozitate, nivelul de umiditate a solului, tipul solului, cantitatea de sol necesară unei plante.

#### *2.1.5. Vizualizarea nevoilor unei plante*

Pentru a avea acces facil și compact la cât mai multe informații esențiale în vederea creșterii plantelor, fiecare plantă oferă informații cu privire la condițiile optime de dezvoltare. Informațiile care pot fi vizualizate pentru fiecare plantă sunt: temperatura aerului, umiditatea aerului, nivelul de luminozitate, nivelul de umiditate a solului, tipul solului, cantitatea de sol necesară.

#### *2.1.6. Afisarea datelor despre plantele din grădina virtuală*

Acest scenariu deservește două posibilități. În cazul în care utilizatorul nu are un sistem intelligent de grădinărit, ci doar acces la aplicația web, acesta poate vedea frecvența cu care a udat planta. În cazul în care utilizatorul are acces și la sistemul de grădinărit intelligent și acesta este configurațat, pentru planta respectivă utilizatorul poate vedea următoarea informație:

- Nivelul de umiditate al aerului
- Nivelul de umiditate al solului
- Temperatura aerului
- Intensitatea luminoasă a mediului

#### *2.1.7. Monitorizarea mediului*

În cadrul acestui scenariu sistemul IOT monitorizează prin intermdiul unor senzori următoarele caracteristici ale mediului plantei:

- Nivelul de umiditate al aerului
- Nivelul de umiditate al solului
- Temperatura aerului
- Intensitatea luminoasă a mediului

În cazul în care una dintre măsurători nu este în parametri optimi, utilizatorul este înștiințat de aceasta la următoarea deschidere a aplicației.

De menționat pe scurt că pentru a avea acces la scenariul de monitorizare este nevoie înainte de completarea unui scenariu auxiliar de configurație a părții hardware a sistemului (scenariul de configurație implică accesul sistemului hardware la rețeaua wi-fi a utilizatorului și detectarea acestuia prin interfață utilizator). Având în vedere că vorbim de un prototip în cazul de față, acest scenariu premergător va fi făcut de către dezvoltatorul prototipului întrucât implică și alterarea codului.

#### *2.1.8. Păstrarea umidității solului adecvată*

O dată configurat, sistemul IOT va păstra umiditatea solului în parametri specificați. Pentru îndeplinirea cu succes a acestui scenariu în momentul în care sistemul rămâne fără apă va informa utilizatorul iar acesta trebuie să acționeze în concordanță, adică să reumple rezervorul de apă al sistemului.

### **2.2. Cerințe non-funcționale**

Pe partea de cerințe non-funcționale sunt atinse următoarele:

- Usability/Utilizabilitatea – interfața web va fi ușor și placut de folosit
- Acessibility/Accesabilitatea – partea de software a proiectului va fi accesibilă publicului larg
- Open source/Acces la codul sursă – pentru a încuraja comunitatea să dezvolte cât mai multe proiecte IOT
- Documentația – astfel încât sistemul să fie ușor de utilizat

## Capitolul 3. Studiu Bibliografic

Pentru această lucrare am investigat soluții atât în agricultura la scară largă cum este prezentată în [3] cât și soluții pentru agricultura la scară micro. Am considerat că relevant pentru aplicația de față sunt sistemele la scară micro.

În lucrarea „Automation using IOT in greenhouse environment” [4] autorii prezintă prototipul unui sistem care monitorizează și controlează diferite aspecte ale unei sere. Ei colecționează informații despre starea mediului din seră folosind senzori. Senzorii măsoară temperatura, intensitatea luminoasă, umiditatea aerului și umiditatea solului. Pentru o bună dezvoltarea a plantelor, autorii folosesc diferite praguri între care păstrează valorile temperaturi și a umidității în seră. Împreună cu o placuță de dezvoltare Neduino 3 Wi-Fi, aceștia regleză temperatura cu ajutorul unor ventilatoare acționate automat și cu ajutorul unor motare care acționează paravane solare în seră pentru. În această lucrare este implementată și o interfață către utilizator în care acesta este notificat de starea serei (măsurările înregistrate de senzori).

S. Aishwarya et al [5] prezintă o soluție de monitorizare a creșterii plantelor de asemenea luând în considerarea temperatura, umiditatea, intensitatea luminoasă dar informația este colectată de la senzori de către o placuță Arduino UNO. O funcționalitate în plus pe care o aduce este monitorizarea stării plantei în sine, nu doar a datelor din seră. Autorii folosesc procesarea de imagini pentru a identifica dacă plantele se dezvoltă adecvat. Deși sera este controlată, există anumite insecte care pot dăuna plantei sau solul poate avea lipsuri de anumite minerale. În ambele cazuri frunzele plantelor vor evidenția această problemă prin anumite comportamente. De exemplu o carență de fier duce la îngălbirea funzelor dar nu și a nervurilor<sup>3</sup>. Pentru a putea monitoriza astfel de scenarii autorii folosesc o cameră de luat vederi și o placuță Raspberry Pi care rulează un algoritm de procesarea a imaginilor ce identifică efectul a diveși dăunători.

Pentru a structura mai bine rezultatul studiului bibliografic în lucrarea de față, vom împărti această secțiune în două mari capitole: partea de proiectarea hardware și partea de proiectarea software. Partea de proiectare hardware constă în componentele fizice care măsoară starea mediului, transmit date, recepționează comenzi și alterează starea mediului supravegheat. Partea de dezvoltarea software face referire la aplicațiile ce furnizează restul funcționalităților menționate anterior. În continuare vom prezenta câteva din opțiunile existente pentru implementarea atât a părții hardware cât și software. De asemenea, tot în acest capitol, vom justifica alegerile făcute.

### 3.1. Proiectarea hardware

În timpul documentării asupra situației actuale din domeniul grădinăritului smart am observat o abordare similară în proiectarea serelor smart, și anume prototipizarea. Majoritatea lucrărilor prezintă un prototip care poate fi extins să funcționeze la scară largă într-o seră de mari dimensiuni. Pentru partea hardware, cea mai importantă decizie este alegerea placuței de dezvoltare care va controla mediul. Secundară acesteia este

---

<sup>3</sup> <https://www.marcoser.ro/consultanta/boli-daunatori-si-probleme-in-culturile-legumicole/carenta-de-fier/>

alegerea senzorilor pe care îi vom folosi și a dispozitivelor care controlează mediul (motoare, pompe, ventilatoare, etc). În continuare vom prezenta opțiunile de dispozitive hardware atât pentru recepționarea datelor și logica de control cât și opțiunile pentru motitorizarea și alterarea mediului.

### 3.1.1. Placa de dezvoltare

În continuare vom prezenta procesul prin care am trecut în alegerea unei plăci de dezvoltare adecvată nevoilor proiectului. S-a dorit ca placa de dezvoltare să poată să citească date de la senzori analogici, să fie ușor de alimentat de la o baterie, să fie fiabilă, să poată fi conectată la internet și să nu fie costisitoare din punct de vedere finanțiar. De asemenea, un aspect care nu a fost neglijat a fost prețul de achiziție. Întrucât lucrarea de față are ca scop creșterea unei plante de apartament în ghiveci, sistemul trebuie să își păstreze costurile într-o limită atractivă pentru utilizator.

Pentru a identifica ce opțiuni există când vorbim de placă de dezvoltare în domeniul serelor smart am recurs la lucrări în domeniu din ultimii zece ani. În timp ce în „An FPGA Based Computer System for Greenhouse Control” [6] autorii folosesc o plăcuță FPGA pentru achiziționarea de date de la senzori și controlul unor paravane în scopul protejării plantelor, [5] folosește o plăcuță Arduino Uno pentru receptionarea datelor și controlul unor ventilatoare și stropitori.

Anterior am menționat și placa de dezvoltare Netduino 3 Wi-Fi care e utilizată în [4]. O altă soluție posibilă este utilizarea unui Raspberry Pi după cum am observat în lucrarea ”IOT Based Smart Greenhouse” [7] în care autorii o folosesc pentru receptionarea datelor și transmiterea lor direct către cloud.

Urmează o scurtă prezentare a fiecărei din plăcuțele mai sus menționate pentru a crea o imagine de ansamblu.

Raspberry Pi prezentat în Figura 3.1 este un minicomputer pe care rulează sistemul de operare Raspbian. El poate fi ușor conectat la internet atât prin Wi-Fi cât și prin Ethernet. Pentru comunicarea cu senzori, Raspberry Pi are nevoie de anumite librării și/sau programe software. Deși nu are spațiu de stocare, el vine cu un slot pentru un card SD.



Figura 3.1 Raspberry Pi 3<sup>4</sup>

---

<sup>4</sup> <https://cleste.ro/raspberry-pi-4-model-b-2gb.html>

Arduino UNO, prezentat în Figura 3.2 e un microcontroler programabil. Limbajul de programare Arduino, specific acestei plăci de dezvoltare, este bazat pe C/C++. Arduino Uno nu poate fi conectată direct la internet ci are nevoie de alte componente hardware (de ex, Wi-Fi shield) pentru a dispune de această funcționalitate. Citirea datelor de la senzori se face foarte ușor cu acesta datorită interfeței simple și pinilor analogici. Arduino UNO vine direct cu spațiu de stocare integrat<sup>5</sup>.



Figura 3.2 Arduino UNO<sup>6</sup>

Netduino este o platformă open-source bazată pe framework-ul .NET micro. Netduino 3 WiFi prezentat în Figura 3.3 este bazat pe microcontrollerul Cortex-M4. El este foarte similar în materie de proiectare cu Arduion, fiind compatibile shiled-uri Arduino cu această placă. WiFi-ul integrat face conectarea la internet foarte ușoară iar framework-ul micro .NET facilitează dezvoltarea programelor cu această placă.

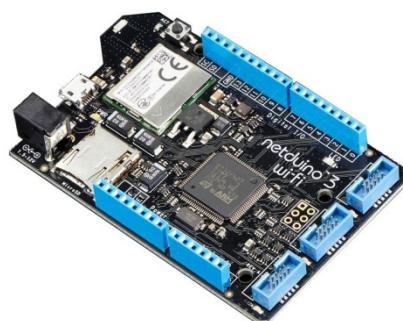


Figura 3.3 Netduino 3 WiFi<sup>7</sup>

---

<sup>5</sup> <https://www.dragonflythingworks.com/2013/11/arduino-vs-netduino-vs-raspberry-pi-vs-beaglebone-one-black-part-1/>

<sup>6</sup> <https://cleste.ro/placa-de-dezvoltare-arduino-uno.html>

<sup>7</sup> <https://www.robofun.ro/net/netduino-3-wifi.html>

Plăcuța FPGA (Field Programmable Gate Array) prezentată în Figura 3.4 este un circuit integrat destinat pentru a fi configurare de către utilizator. Limbajul specific de programare a aceste plăci este HDL. Pentru conexiunea la internet aceasta necesită atât alte piese hardware cât și software specific. Citirea datelor de la senzori implică și ea mai multă configurare a plăcuței.

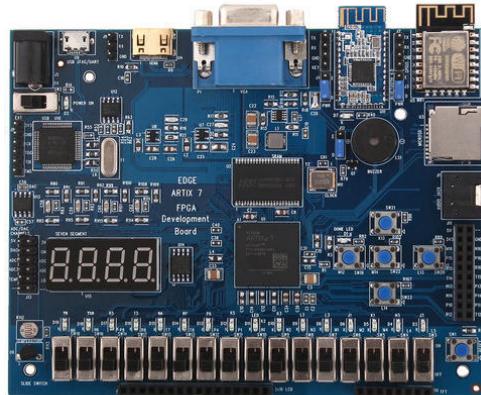


Figura 3.4 Plăcuță FPGA<sup>8</sup>

Din opțiunile studiate și prezentate până acum, alegerea mea a fost plăcuță Netduino 3 WiFi dar în timpul căutării acestieia de la furnizorii, am descoperit o alternativă recentă a elementului de control, și anume plăcuța UNO + Wi-Fi R3 care conține o integrare completă a microcontrolerului Atmel ATmega328 și a Wi-Fi-ului ESP8266 – Figura 3.5. Ea este programabilă prin Arduino, are o interfață ușor de utilizat în interacțiunea cu elementele periferice și bineînțeles poate fi conectată cu ușurință la internet.

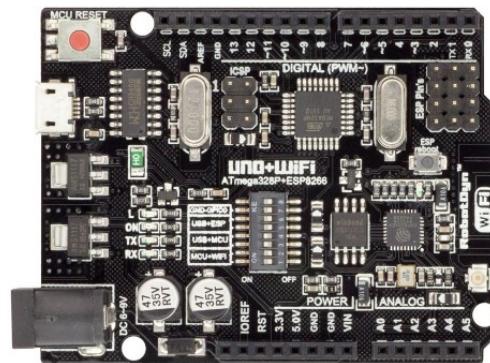


Figura 3.5 UNO + Wi-Fi R3<sup>9</sup>

---

<sup>8</sup> <https://www.distrelec.ro/ro/fpga-board-uart-usb-digilent-410-183/p/30044247>

<sup>9</sup> <https://cleste.ro/atelier/placa-dezvoltare-unowifi-r3/>

Pentru alegerea plăcuței cu care să proiectăm sistemul de grădinărit inteligent am evidențiat în Tabel 3.1 Comparare între dispozitivele hardware de control nivelul de dificultate cu care o putem programa, cât de ușor putem citi date de la senzori, simplitatea cu care ne putem conecta la internet cât și prețul de achiziție.

Tabel 3.1 Comparare între dispozitivele hardware de control

Plăcuță	Ușor și rapid programabil	Poate fi conectată la internet fără alte resurse	Poate citi date direct de la senzori analogici	Preț de achiziție scăzut
Raspberry Pi	Da	Da	Nu	Nu <sup>10</sup>
Arduino	Da	Nu	Da	Da <sup>11</sup>
Netduino	Da	Da	Da	Nu <sup>12</sup>
FPGA	Nu	Nu	Nu	Nu <sup>13</sup>
Uno + Wi-Fi R3	Da	Da	Da	Da <sup>14</sup>

Observăm că Uno + Wi-Fi R3 este singura opțiune care răspunde cu da cerințelor noastre. Fiind o opțiune cu modul de Wi-Fi integrat, aceasta reduce și posibilitate defectării componentelor auxiliare cum ar fi în cazul Arduino Uno. Este cu mult mai ieftină decât Raspberry Pi și de asemenea citirea datelor de la senzorii analogici se face cu o mai mare ușurință. Având în vedere scopul lucrării, microcontrolerul Uno + Wi-Fi R3 este cea mai potrivită alegere.

### 3.1.2. Dispozitive periferice

În alegerea dispozitivelor periferice ne referim atât la cele care măsoara date despre mediu cât și la cele care alterează mediu.

Dispozitivele care măsoară date despre mediu sunt senzorii. Pentru a alege senzorii potriviti vom lua în considerare tipul de informație pe care aceștia o furnizează, modul de transmitere a datelor, și prețul de achiziție. În majoritatea lucrărilor studiate [4], [5], [6], [7] senzorii predominați sunt:

- Senzor de temperatură a aerului
- Senzor de umiditate a aerului
- Senzor de umiditate a solului
- Senzor de intensitate luminoasă sau fotorezistor

Având în vedere că datele furnizate de acești senzori sunt suficiente pentru monitorizarea unei sere, am considerat că acești senzori pot măsura starea de bine a plantei și în cazul lucrării de față. Deoarece niciunul dintre aceștia nu are un preț de achiziție ridicat<sup>15 16 17</sup> am decis să îi folosim pe toți în lucrarea de față.

<sup>10</sup> <https://cleste.ro/raspberry-pi-4-model-b-2gb.html>

<sup>11</sup> <https://cleste.ro/placa-de-dezvoltare-arduino-uno.html>

<sup>12</sup> <https://www.robofun.ro/net/netduino-3-wifi.html>

<sup>13</sup> <https://www.distrelec.ro/ro/fpga-board-uart-usb-digilent-410-183/p/30044247>

<sup>14</sup> <https://cleste.ro/atelier/placa-dezvoltare-unowifi-r3/>

<sup>15</sup> <https://cleste.ro/senzor-digital-de-temperatura-i-umiditate-dht11-cu-led.html>

Dispozitivele care controlează mediul folosite în lucrările prezentate anterior sunt de tipul

- Ventilatoarea care reglează fluxul de aer
- Motoare care deplasează paravane solare (draperii)
- Pompe de apă pentru irigare.

Întrucât lucrarea de față își propune creșterea unei plante în ghiveci, nu în seră, modificare stării întregului apartament în care se află ghiveciul nu e fezabilă aşa încât singurul dispozitiv de alterare a mediului pe care îl vom folosi dintre acestea este pompa de apă.

### 3.2. Proiectare software

Partea software a lucrării va deservi atât utilizatorii care dețin și partea hardware dar și pe cei care vor să utilizeze doar parte software. Pentru a servi nevoile utilizatorilor am ales să analizăm aplicații existente pe piață și nu lucrări științifice. Motivul acestei alegeri este că lucrările științifice nu au un grup de utilizatori larg care își poate spune părerea despre cât gradul de utilitate al proiectului, în timp ce aplicațiile existente dețin nenumărate păreri și recomandări din partea utilizatorilor.

Prima categorie de aplicații pe care am studiat-o este cea de identificare a plantelor. Aceasta sunt folositoare în cazul în care utilizatorul nu știe ce plantă deține și nu cunoaște nici nevoile acesteia. Aplicație pe care o prezentăm este „Garden Answers Plant Identification”<sup>18</sup>. Este vorba de o aplicație gratuită a cărei funcționalitate este de a identifica ce fel de plantă deține utilizatorul. Modul de utilizare al aplicației este următorul: utilizatorul face o poză plantei sale, o încarcă în aplicație iar aceasta sugerează o listă de opțiuni pe care le consideră adecvate. Utilizatorul trebuie să selecteze rezultatul care se potrivește cel mai mult cu planta sa. Ultima parte a acestui scenariu se poate observa în figura 3.6.

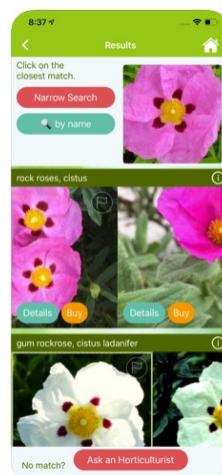


Figura 3.6 Garden Answers Plant Id – selectarea plantei

---

<sup>16</sup> <https://cleste.ro/modul-cu-senzor-umiditate-sol.html>

<sup>17</sup> <https://cleste.ro/fotorezistor-5528-ldr.html>

<sup>18</sup> <https://apps.apple.com/us/app/garden-answers-plant-identification/id605855033>

Cea de a doua categorie de aplicații din cele studiate pe care am ales să le observăm mai în detaliu este cea de păstrare a evidenței plantelor și regularitatea cu care acestea sunt udate. În această categorie intră aplicații precum „Waterbot: Plants watering + Gardening”<sup>19</sup> și „Vera: Plant Care App”<sup>20</sup>. Cea din urmă, de asemenea gratuită ca și „Garden Answers Plant Identification”<sup>21</sup>, se axează pe udarea regulată a plantelor. Ea oferă posibilitatea utilizatorului să își creeze câte un profil pentru fiecare plantă pe care o deține – se poate observa în figura 3.7 – și să să seteze pentru fiecare plantă un program de udare în funcție de nevoile acesteia. Astfel, utilizatorul va avea un portofoliu de plante și va primi, separat, pentru fiecare plantă, o notificare în ziua în care aceasta trebuie udată. Aplicația ajută utilizatorul atât să își mențină plantele în viață prin udarea lor regulată cât și oferă o privire de ansamblu asupra plantelor pe care acesta le deține.

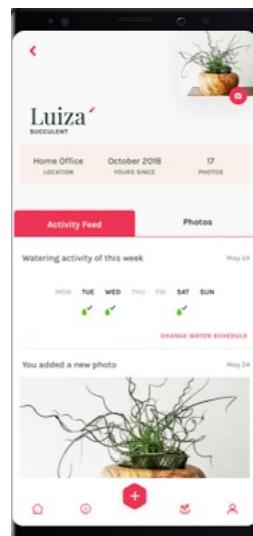


Figura 3.7 Vera – profilul unei plante

Folosind aplicațiile prezentate ca bază de pornire, lucrarea de față își propune să creeze o aplicație care să ofere componenta de monitorizare și gestionare a plantelor și a intervalului de udare sau a reglării umidității solului. Funcționalitatea de identificarea a plantei sau a nevoilor acestora poate prezenta o dezvoltare ulterioară. Pentru prototipul actual am decis să ne axăm pe monitorizarea și creșterea plantelor cunoscute.

<sup>19</sup> <https://play.google.com/store/apps/details?id=net.kosev.watering&hl=en>

<sup>20</sup> <https://www.veraplantcareapp.com>

<sup>21</sup> <https://apps.apple.com/us/app/garden-answers-plant-identification/id605855033>

## Capitolul 4. Analiză și Fundamentare Teoretică

Considerând obiectivele și specificațiile menționate în capitolul 2, putem împărți aplicația în două părți. Luăm această decizie deoarece putem oferi o imagine mai cuprinzătoare și mai ușor de înțeles și analizat dacă luăm în considerare aceste părți componente pe rând. În Figura 4.1 se observă că partea hardware se referă la componentele care monitorizează, controlează și alterează mediul plantei iar partea software se referă la aplicația web disponibilă utilizatorilor. Când vorbim de partea software includem atât aplicația server cât și cea client.

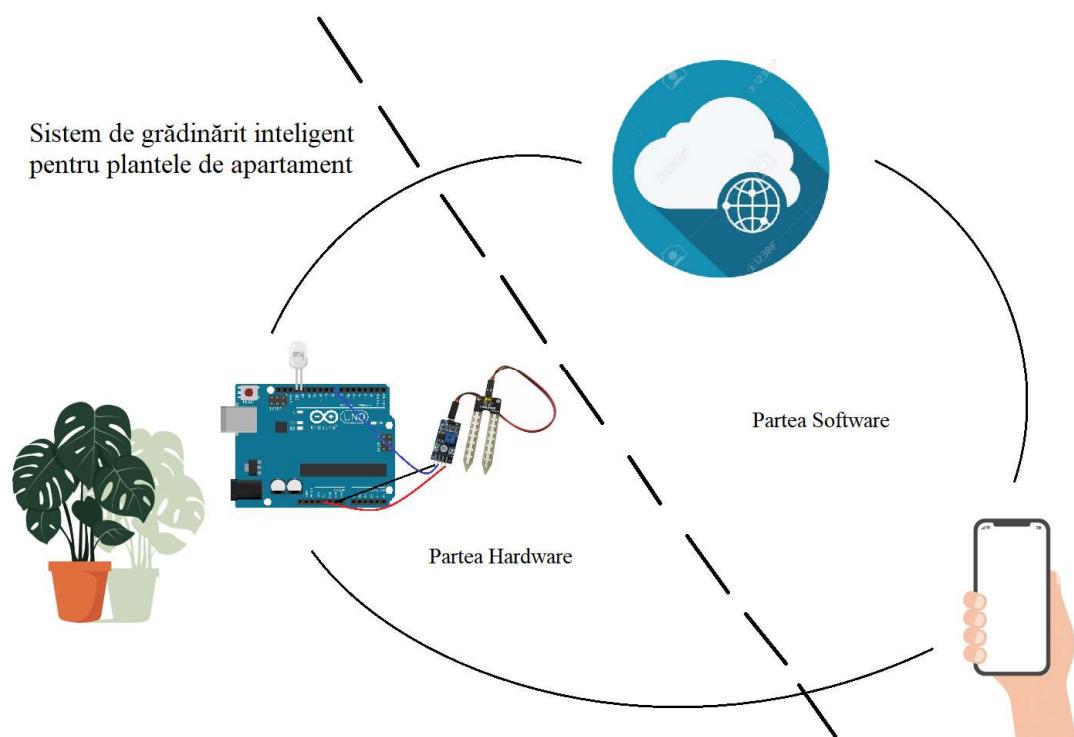


Figura 4.1 Arhitectura conceptuală a sistemului

Prima parte este partea software, accesibilă gratuit oricărui utilizator. Aceasta are ca scop evidența plantelor din apartament și gestionarea programului de udare a acestora. Principalul flux al părții software este reprezentat în Figura 4.2. Utilizatorul începe prin a vizualiza lista de plante disponibile în aplicație și a selecta una dintre aceste plante. Selectarea unei plante implică adăugarea acesteia în grădina virtuală a utilizatorului. Pasul următor este configurarea sistemului hardware. A se menționa că din cei cinci mari pași descriși, pașii numărul trei și patru sunt facultativi. Acești pași au sens doar pentru utilizatorii care beneficiază și de parte hardware a aplicației. Ultimul pas este reprezentat de vizualizare datelor despre plantă. Acestea pot fi date și despre starea mediului plantei în cazul în care utilizatorul beneficiază de parte hardware.

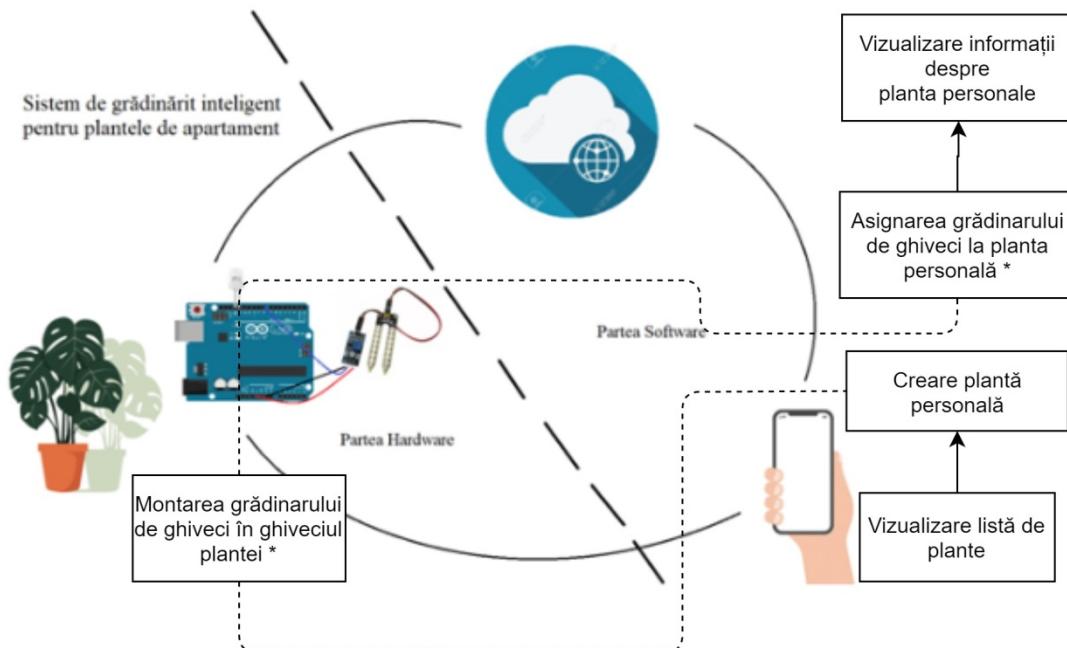


Figura 4.2 Fluxul de business al părții software a aplicației

A doua parte a aplicației este parte de hardware. Aceasta are ca scop oferirea unui mediu supravegheat și parțial controlat de creștere a plantei. Fluxul de lucru al părții hardware este evidențiat în Figura 4.3. Se poate observa că primul pas e cel de configurare. După configurare, parte hardware a aplicației intră într-o buclă de măsurare și evaluare (conform configurației) a mediului plantei. În cazul în care e nevoie de ajustare automată, sistemul va ajusta singur condițiile de mediu. În cazul în care este nevoie de ajutare manuală, sistemul va notifica utilizatorul cu privire la ce acțiuni trebuie acesta să întreprindă pentru a oferi plantei un mediu propice de dezvoltare.

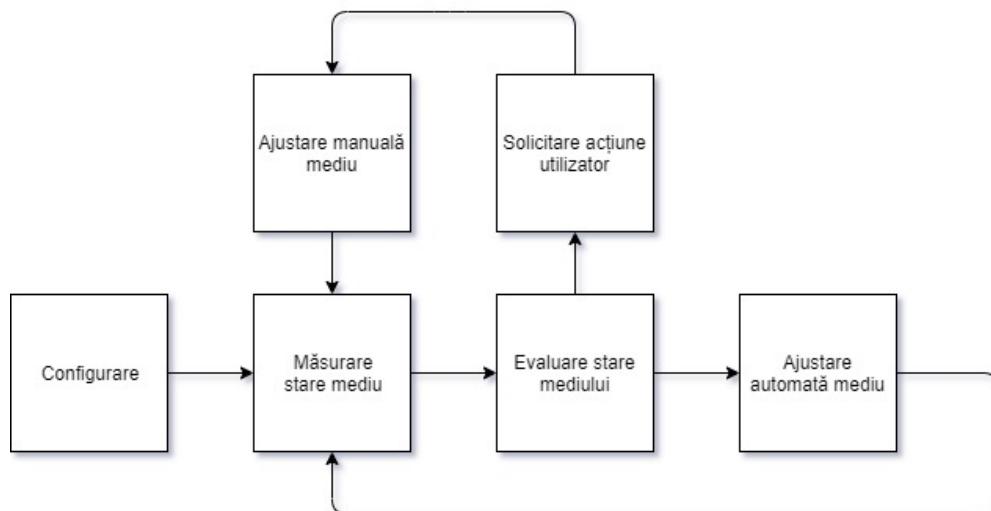


Figura 4.3 Fluxul de lucru al părții hardware a aplicației

O privire mai în detaliu asupra aplicației, pe lângă fluxurile principale ale acesteia o oferă cazurile de utilizare ale aplicației. Spre deosebire de cerințele funcționale absolut necesare prezentate în capitolul 2, în urma unei analize mai în detaliu am decoperit nevoie de anumite cerințe funcționale adjuvante pentru aplicația noastră.

În subcapitolul următor vom prezenta în detaliu cazurile de utilizare pentru fiecare cerință funcțională.

#### 4.1. Cazuri de utilizare

În capitolul 2 am prezentat cerințele funcționale ale aplicației. Acestea le-am analizat mai în detaliu și au rezultat cazurile de utilizare ale aplicației. Ele sunt prezentate în Figura 4.4.

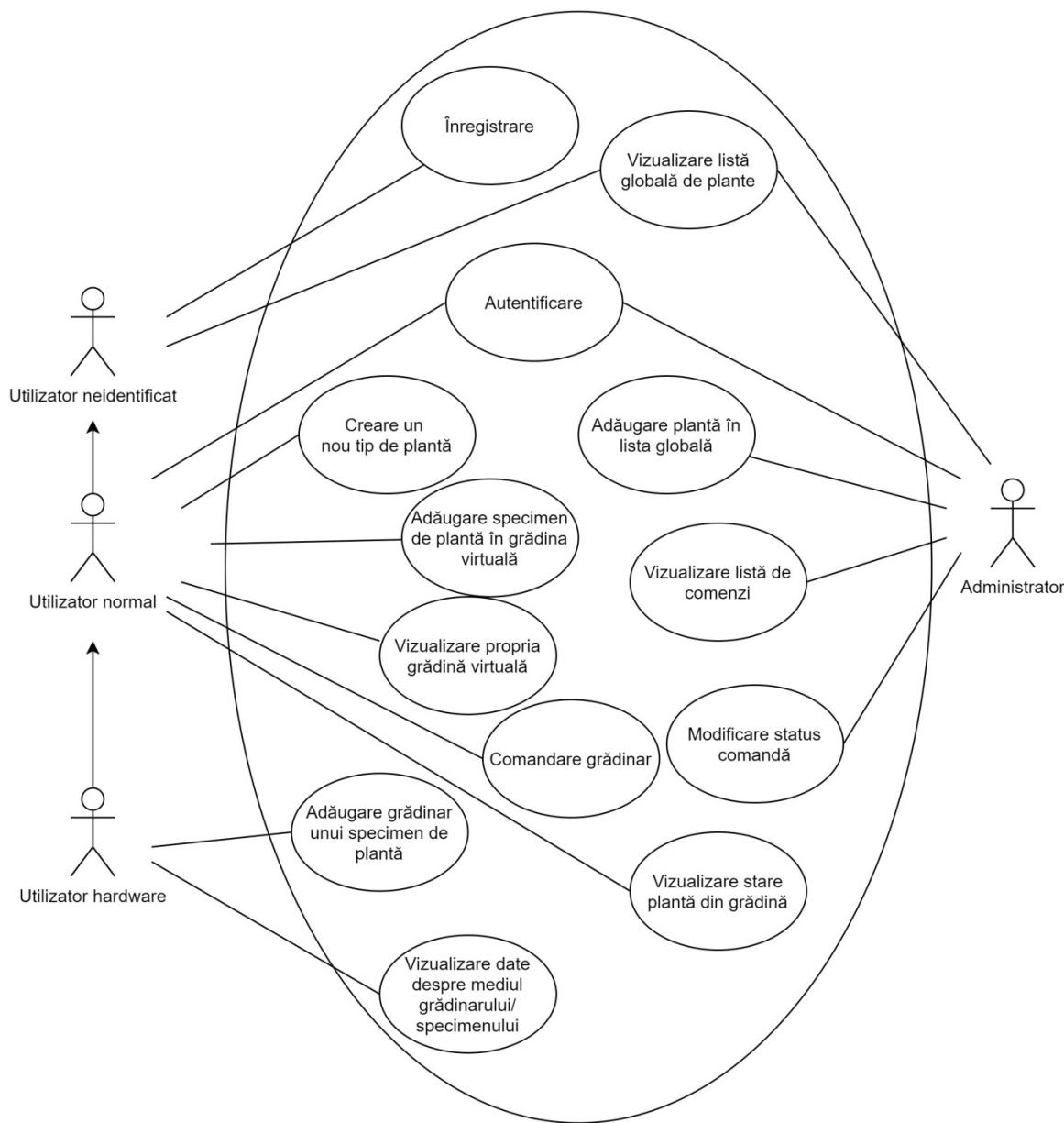


Figura 4.4 Cazuri de utilizare a aplicație

Observăm în Figura 4.4 cazurile de utilizare ce vor fi detaliate ulterior și patru roluri:

- Utilizator neautentificat – rolul este reprezentativ pentru vizitorii aplicatiei software care nu dețin un cont
- Utilizator normal – rolul e reprezentativ pentru persoana care are un cont în aplicație
- Utilizator hardware – rolul este reprezentativ pentru persoana care are un cont în aplicație și deține și partea hardware a sistemului. Acest rol extinde roulul de utilizator normal
- Administrator – rolul e reprezentat de persoanele care administrază lista globală de plante a aplicației.

#### 4.1.1. Înregistrare

Cazul de utilizare este prezentat în tabelul 4.1. De menționat că aplicația poate fi utilizată și fără un cont, dar crearea acestuia oferă mai multe funcționalități utilizatorului.

Tabel 4.1 Caz de utilizare – Înregistrarea unui utilizator în aplicație

<b>Descrierea cazului de utilizare</b>	Utilizatorul își face cont în cadrul aplicației pentru a beneficia de mai multe funcționalități.
<b>Actori</b>	Utilizator normal
<b>Precondiții</b>	Acces la internet Acces la aplicație
<b>Postcondiții</b>	Un nou cont a fost creat în aplicație. Utilizatorul are acces la aplicație cu numele de utilizator specificat în formular. Utilizatorul are acces la aplicație cu parola specificată în formular. Administratorul poate vedea noul utilizator în lista de utilizatori.
<b>Flux principal</b>	<ol style="list-style-type: none"> <li>1. Utilizatorul accesează pagina de înregistrare</li> <li>2. Utilizatorul își introduce adresa de e-mail</li> <li>3. Utilizatorul își introduce parola</li> <li>4. Utilizatorul confirmă parola</li> <li>5. Utilizatorul apasă butonul „Înregistrare”</li> </ol>
<b>Extensi</b>	<p>2a_1. Utilizatorul nu a introdus o adresă de e-mail validă.</p> <p>2a_2. Utilizatorul este informat de către sistem că adresa de e-mail trebuie să fie într-un format valid</p> <p>3a_1. Utilizatorul introduce o parolă mai scurtă de 6 caractere.</p> <p>3a_2. Utilizatorul este informat de către sistem că trebuie să folosească o parolă de cel puțin 6 caractere</p> <p>4a_1. Utilizatorul nu a introdus aceeași parolă în câmpul de confirmare a parolei</p> <p>4a_2. Utilizatorul este informat de către sistem că trebuie să introducă</p>

	<p>același text în câmpul de introducere a parolei și în cel de confirmare a acesteia</p> <p>5a_1. Un utilizator este deja înregistrat în aplicație cu această adresă de e-mail.</p> <p>5a_2. Utilizatorul este informat de către sistem să verifice adresa de e-mail folosită iar în cazul în care aceasta este corectă să ceară resetarea parolei.</p>
--	--

#### 4.1.2. Autentificare

Cazul de utilizare pentru autentificarea utilizatorilor în aplicație este prezentat în tabelul 4.2. De menționat că doar utilizatorii normali au nevoie să se înregistreze în prealabil. În cazul administratorilor, conturile acestora sunt create manual în sistem.

Tabel 4.2 Caz de utilizare – Autentificarea utilizatorilor în aplicație

<b>Descrierea cazului de utilizare</b>	Utilizatorul se autentifică în aplicație pentru a beneficia de mai multe funcționalități.
<b>Actori</b>	Utilizatorul normal Administratorul
<b>Precondiții</b>	Acces la internet Acces la aplicație Utilizatorul normal și-a creat cont în aplicație
<b>Postcondiții</b>	Utilizatorul este autentificat în aplicație. Utilizatorul este redirectionat spre o pagină specifică.
<b>Flux principal</b>	<ol style="list-style-type: none"> <li>1. Utilizatorul accesează pagina de autentificare</li> <li>2. Utilizatorul introduce adresa de e-mail în câmpul corespunzător</li> <li>3. Utilizatorul introduce parola în câmpul corespunzător</li> <li>4. Utilizatorul apasă butonul de autentificare</li> </ol>
<b>Extensi</b>	<p>2a_1. Utilizatorul nu a introdus o adresă de e-mail validă.</p> <p>2a_2. Utilizatorul este informat de către sistem că adresa de e-mail trebuie să fie într-un format valid</p> <p>4a_1. Utilizatorul a introdus o adresă de e-mail greșită.</p> <p>4a_2. Utilizatorul este informat de către sistem că această adresă de e-mail nu figurează cu un cont în aplicație</p> <p>4b_1. Utilizatorul a introdus o parolă greșită.</p> <p>4b_2. Utilizatorul este informat de către sistem că a introdus o parolă greșită și este invitat să corecteze parola sau să recurgă la funcționalitatea de resetarea a parolei</p>

#### 4.1.3. Vizualizare listă globală de plante

Cazul de utilizare este descris în tabelul 4.4. Lista global disponibilă de plante despre care aplicația deține informații. Nu este o listă exhaustivă de plante. De menționat că acesta este singurul caz pentru care nu este nevoie de autentificare.

Tabel 4.3 Caz de utilizare – Vizualizare listă globală de plante

<b>Descrierea cazului de utilizare</b>	Vizualizarea listei de plante din aplicație pentru identificarea nevoilor unei plante
<b>Actori</b>	Utilizator normal Utilizator neautentificat
<b>Precondiții</b>	Acces la internet Acces la aplicație
<b>Postcondiții</b>	Acces la pagina cu lista globală de plante
<b>Flux principal</b>	1. Utilizatorul accesează aplicația 2. Utilizatorul accesează lista globală de plante
<b>Extensiile</b>	

#### 4.1.4. Adăugare plantă în lista globală

Având în vedere statutul de prototip al aplicației, în fază incipientă, lista globală de plante nu va fi numeroasă. Pentru ca aceasta să poată fi extinsă în timp există funcționalitatea de adăugare de noi plante de către un administrator. Cazul de utilizare este descris în tabelul 4.5.

Tabel 4.4 Caz de utilizare – Adăugare plantă în lista globală

<b>Descrierea cazului de utilizare</b>	Adăugare plantă în lista globală de plante pentru extinderea acesteia
<b>Actori</b>	Administrator
<b>Precondiții</b>	Acces la internet Acces la aplicație Actorul este autentificat în aplicație
<b>Postcondiții</b>	O nouă plantă este adăugată în lista globală Planta adăugată este vizibilă în lista globală fără autentificare
<b>Flux principal</b>	1. Administratorul accesează lista globală de plante 2. Administratorul apasă butonul de adăugare o nouă plantă 3. Administratorul este redirecționat spre pagina de completare a formularului cu date despre plantă 4. Administratorul completează câmpurile de date cu informații relevante despre planta în cauză 5. Administratorul salvează noua plantă 6. Administratorul este redirecționat spre lista globală de plante
<b>Extensiile</b>	4a 1. Administratorul nu a completat un câmp de introducere a datelor

	4a_2. Administratorul este informat de către aplicație că toate câmpurile sunt obligatorii.
--	---

#### 4.1.5. Vizualizare grădina virtuală

Deoarece scopul aplicației este de a ajuta utilizatorul să își crească plantele personale, o imagine de ansamblu asupra acestora în cadrul aplicației este absolut necesară. În grădina virtuală utilizatorul poate vedea toate plantele pe care acesta le-a adăugat pe parcursul timpului. Ea este disponibilă doar pentru utilizatorul în cauză, niciun alt utilizator nu este capabil să vadă altă grădina virtuală decât a sa.

Tabel 4.5 Caz de utilizare – Vizualizare grădina virtuală

<b>Descrierea cazului de utilizare</b>	Utilizatorul dorește să vizuleze grădina sa virtuală
<b>Actori</b>	Utilizator normal
<b>Precondiții</b>	Acces la internet Acces la aplicație Utilizatorul este autentificat în aplicație
<b>Postcondiții</b>	Grădina virtuală este afișată
<b>Flux principal</b>	1. Utilizatorul accesează aplicația 2. Utilizatorul este redirecționat spre pagina grădinii virtuale
<b>Extensiile</b>	

#### 4.1.6. Adăugare plantă în grădina virtuală

Funcționalitatea de adăugare plantă în grădina virtuală deservește nevoia utilizatorului de a-și crea grădina virtuală cu o schemă a plantelor reprezentativă situației din viață reală. Cazul de adăugare a unei plante în grădina virtuală este prezentat în tabelul 4.7.

Tabel 4.6 Caz de utilizare – Adăugare plantă în grădina virtuală

<b>Descrierea cazului de utilizare</b>	Utilizatorul dorește să își adauge o plantă în grădina virtuală din cadrul aplicației.
<b>Actori</b>	Utilizator normal
<b>Precondiții</b>	Acces la internet Acces la aplicație Utilizatorul este autentificat în aplicație
<b>Postcondiții</b>	Planta a fost adăugată în aplicație în cadrul grădinii virtuale a utilizatorului. Planta este de tipul selectat (sau configurat) de către utilizator. Planta este vizibilă în grădina virtuală a utilizatorului. Planta nu este vizibilă pentru oricare alt utilizator.

<b>Flux principal</b>	<ol style="list-style-type: none"> <li>1. Utilizatorul acceseează grădina virtuală din aplicație</li> <li>2. Utilizatorul apasă butonul de adăugare o nouă plantă</li> <li>3. Utilizatorul este redirecționat spre pagina de adăugare plantă</li> <li>4. Utilizatorul introduce date specifice care să identifice exemplarul în grădina sa virtuală.</li> <li>5. Aplicația afișează lista de tipuri de plante existente (din lista globală de plante)</li> <li>6. Utilizatorul caută prin lista de tipuri de plante existente</li> <li>7. Utilizatorul identifică tipul de plantă pe care vrea să o adauge</li> <li>8. Utilizatorul selectează tipul de plantă</li> <li>9. Utilizatorul apasă butonul de adăugare în grădina virtuală</li> <li>10. Utilizatorul este redirecționat spre grădina sa virtuală</li> </ol>
<b>Extensiile</b>	<p>7_1. Utilizatorul nu identifică în lista globală de plante niciun tip de plantă care se pretează pentru exemplarul său</p> <p>7a_2. Utilizatorul va crea un tip de plantă local (vizibil doar pentru el)</p> <p>7a_3. Noul tip de plantă este disponibil în lista cu tipuri de plantă existente (doar pentru utilizatorul în cauză)</p> <p>7a_4 Utilizatorul reia scenariul de la pasul 1</p>

#### 4.1.7. Creare tip de plantă local

Întrucât aplicația nu deține o listă exhaustivă de plante, unii utilizatori e posibil să nu găsească în lista globală de plante un tip de plantă adecvat exemplarului pe care aceștia îl dețin. În acest caz, oferim utilizatorilor posibilitatea ca aceștia să creeze un nou tip de plantă în cadrul grădinii lor virtuale pe care să îl poată utiliza pentru adăugarea de noi exemplare.

Tabel 4.7 Caz de utilizare – Creare plantă configurată manual

<b>Descrierea cazului de utilizare</b>	Creare tip de plantă configurabil manual
<b>Actori</b>	Utilizator normal
<b>Precondiții</b>	Acces la internet Acces la aplicație Utilizatorul este autentificat
<b>Postcondiții</b>	Un nou tip de plantă a fost creat Tipul de plantă este disponibil utilizatorului care l-a creat Tipul de plantă nu este disponibil altor utilizatori
<b>Flux principal</b>	<ol style="list-style-type: none"> <li>1. Utilizatorul acceseează lista globală de plante</li> <li>2. Utilizatorul apasă butonul de adăugare tip de plantă</li> <li>3. Aplicația afișează secțiunea formularului de creare tip de plantă</li> <li>4. Utilizatorul completează câmpurile de date din formular.</li> <li>5. Utilizatorul salvează noul tip de plantă</li> </ol>
<b>Extensiile</b>	<p>4a_1. Utilizatorul nu a completat un câmp de date obligatoriu</p> <p>4a_2. Utilizatorul este informat de către aplicație că toate câmpurile de</p>

date sunt obligatorii
-----------------------

#### 4.1.8. Vizualizare informații plantă din grădină

Vizualizarea informațiilor unei plante din grădina virtuală se adresează și utilizatorilor care nu au acces la partea hardware a aplicației. Aici se regăsesc informații despre necesitățile acestui exemplar în funcție de tipul plantei și informațiile specifice pe care utilizatorul le-a salvat despre aceasta. Informații specifice se referă la dimensiunea plantei, anotimpul curent, luminozitatea de care aceasta beneficiază. Cazul de utilizare este prezentat în tabelul 4.10.

Tabel 4.8 Caz de utilizare – Vizualizare informații plantă din grădină

<b>Descrierea cazului de utilizare</b>	Utilizatorul dorește să vizualizeze informații despre un exemplar din grădina sa virtuală
<b>Actori</b>	Utilizator normal
<b>Precondiții</b>	Acces la internet Acces la aplicație Utilizatorul este autentificat în aplicație Utilizatorul deține cel puțin un exemplar în grădina sa virtuală
<b>Postcondiții</b>	Aplicația afișează informații despre exemplarul pe care l-a selectat utilizatorul
<b>Flux principal</b>	<ol style="list-style-type: none"> <li>1. Utilizatorul navighează la pagina grădinii virtuale.</li> <li>2. Utilizatorul apasă pe unul dintre exemplarele din grădina virtuală</li> <li>3. Aplicația afișează informații relevante cu privire la exemplarul selectat de utilizator</li> </ol>
<b>Extensii</b>	

În Figura 4.5 am evidențiat printr-o diagramă de flux cazurile de utilizare pentru

- înregistrare,
- autentificare,
- vizualizare listă de plante,
- adăugare tip de plantă,
- adăugare specimen de plantă și
- vizualizarea nevoilor acestuia în grădina utilizatorului.

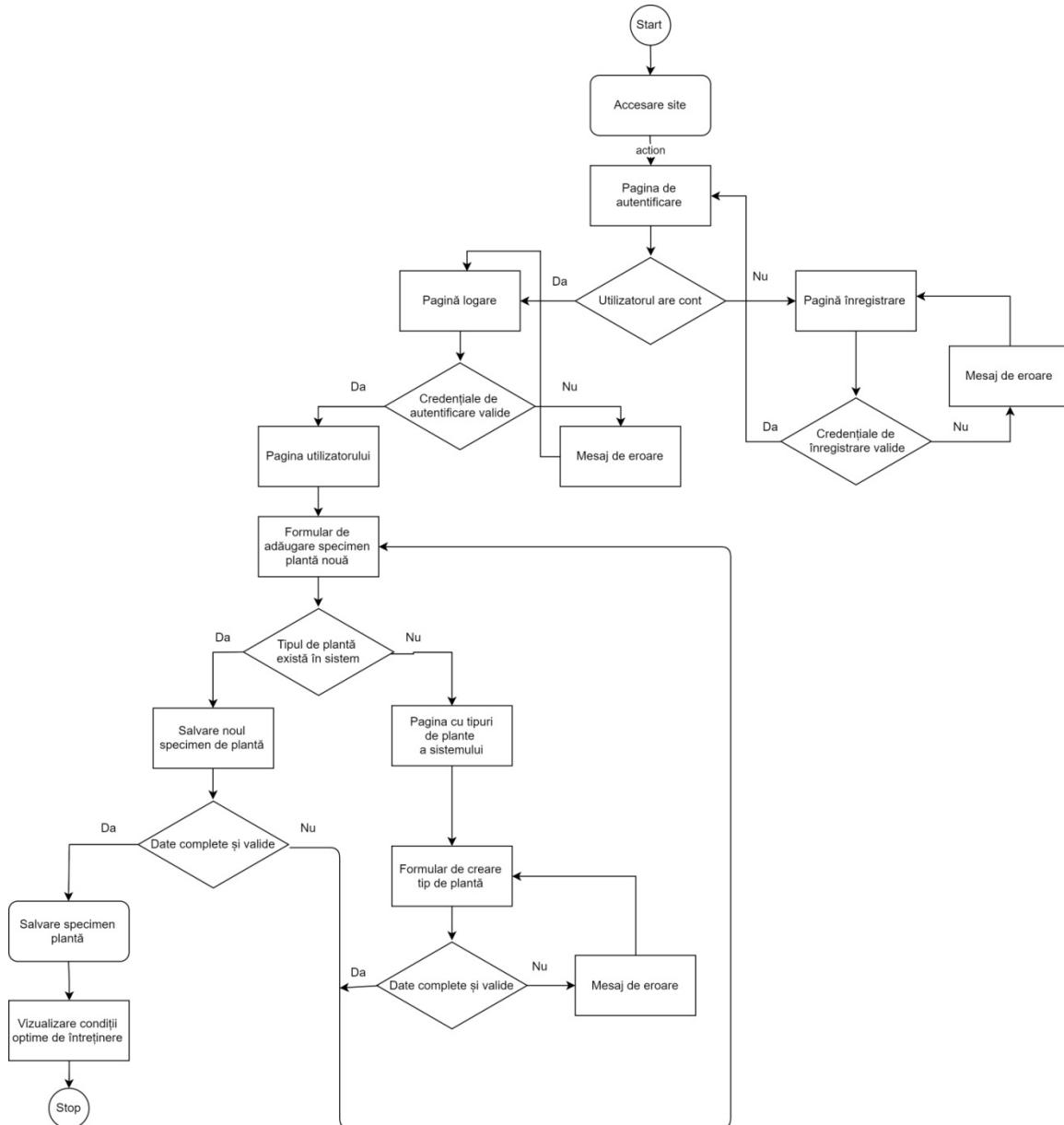


Figura 4.5 Diagramă de flux pentru crearea unui specimen nou (include toate cazurile de utilizare care sunt postcondiții pentru acesta)

#### 4.1.9. Proces de achiziție a Grădinariului de ghiveci (modulul hardware)

Această funcționalitate se adresează doar utilizatorilor care doresc să aibă acces și la partea hardware a aplicației. De menționat că partea hardware furnizată unui utilizator este configurată din fabrică special pentru contul utilizatorului respectiv. Un utilizator poate avea mai multe părți hardware care aparțin aceluiași cont (pentru mai multe plante din apartament) dar o componentă hardware nu poate monitoriza două plante și nici nu poate fi accesată din două conturi. Cazul de utilizare pentru comandarea unui Grădinar de ghiveci este prezentat în tabelul 4.11.

Tabel 4.9 Caz de utilizare – 4.1.9. Proces de achiziție a grădinarului de ghiveci (modulul hardware)

<b>Descrierea cazului de utilizare</b>	Utilizatorul dorește să configureze partea hardware a aplicației după nevoile plantei
<b>Actori</b>	Utilizator hardware Administrator
<b>Precondiții</b>	Acces la internet Acces la aplicație Utilizatorul este autentificat în aplicație
<b>Postcondiții</b>	Partea hardware a sistemului este configurată conform specificațiilor și livrată utilizatorului. Parte hardware este reprezentată în sistem sub forma unei entități de tipul „Gardener”
<b>Flux principal</b>	<ol style="list-style-type: none"> <li>1. Utilizatorul accesează pagina „Grădina mea”</li> <li>2. Utilizatorul apasă butonul de comandare grădinar.</li> <li>3. Utilizatorul introduce datele necesare.</li> <li>4. Sistemul crează o comandă în statusul „PENDING”</li> <li>5. Un administrator acceptă comanda</li> <li>6. Comanda ajunge în statusul „ACCEPTED”</li> <li>7. Administratorul și utilizatorul au o discuție live în care stabiliesc specificațiile ghiveciului cu care vine grădinarul semiautomat</li> <li>8. Administratorul trimite prin curier ghiveciul cu grădinar utilizatorului</li> <li>9. Administratorul actualizează comanda în statusul „COMPLETED”</li> <li>10. Sistemul crează o reprezentarea a grădinarului</li> <li>11. Utilizatorul poate vizualiza noul Grădinar de ghiveci în pagina sa</li> </ol>
<b>Extensi</b>	<ol style="list-style-type: none"> <li>3a_1. Datele introduse de utilizator nu sunt complete.</li> <li>3a_2. Sistemul afișează un mesaj de eroare.</li> <li>3a_3. Utilizatorul corectează datele și reia de la pasul 3.</li> </ol>

#### 4.1.10. Vizualizare date despre mediul planetei

Acest scenariu se adresează doar utilizatorilor care au și dispozitivul hardware. Cazul de utilizare este prezentat în tabelul 4.12.

Tabel 4.10 Caz de utilizare – Vizualizare date despre mediul planetei

<b>Descrierea cazului de utilizare</b>	Utilizatorul dorește să vadă informații despre mediul în care se află planta și dacă acestea sunt în parametri normali pentru dezvoltarea optimă a plantei.
<b>Actori</b>	Utilizator hardware
<b>Precondiții</b>	Acces la internet Acces la aplicație Utilizatorul deține modulul hardware al aplicației Modulul hardware al aplicației este configurat

<b>Postcondiții</b>	Afișarea stării mediului și a analizei acesteia
<b>Flux principal</b>	<ol style="list-style-type: none"> <li>Utilizatorul accesează pagina contului său</li> <li>Utilizatorul vizualizează date înregistrate de grădinarul semiautomat</li> <li>Aplicația afișează utilizatorului ultimele valori înregistrate de către modulul hardare și cât de benefice sunt considerate acestea pentru exemplarul configurat.</li> </ol>
<b>Extensiî</b>	

În Figura 4.6 am evidențiat printr-o diagramă de acțiuni cazurile de utilizare 4.1.9 și 4.1.10 pentru procesul de achiziție a unui grădinar semiautomat și vizualizarea datelor înregistrate de acesta.

Observăm că acest caz de utilizare are nevoie de doi actori pentru a fi îndeplinit. Aceasta este din cauză că o comandă pentru un grădinar de grădină vine și cu o componentă de interacțiune în mod direct cu viitorul utilizator. Scopul acestei întâlniri este stabilirea ghiveciului în care va fi instalat grădinarul inteligent.

Întrucât vrem ca să deservim cât mai multe tipuri și dimensiuni de plante venim în întâmpinarea utilizatorilor cu această opțiune de a stabili împreună cu aceștia dimensiunea necesară a ghiveciului, tipul de plantă care s-ar portriui cel mai bine în apartamentul acestora în funcție de condițiile de lumină și nu în ultimul rând, decorul ghiveciului pentru un plus de personalitate.

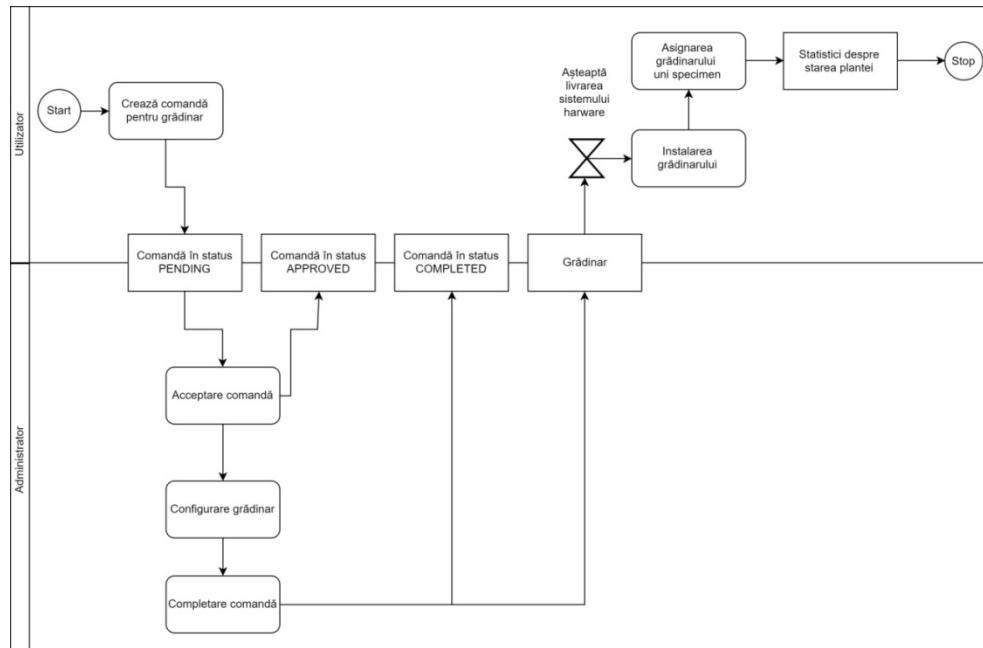


Figura 4.6 Proces de achiziție a unui Grădinar de ghiveci și vizualizarea datelor furnizate de acesta

## 4.2. Module

După prezentarea cazurilor de utilizare ale aplicației și o scurtă analiză a acestora, aplicația poate fi împărțită în patru module principale:

- Modulul de gestionarea a utilizatorilor
- Modulul de gestionare a plantelor
- Modulul de grădinărit semi-automat

Am ales această împărțire deoarece izolează nu doar tipul obiectelor pe care fiecare modul le gestionează, dar și motivele pentru care fiecare modul s-ar putea schimba. Pentru a respecta principul responsabilității unice („Single Responsibility Principle”) [8] luăm în considerare motivele pentru care poate fi nevoie ca un modul să își schimbe funcționalitățile. Astfel, dacă vom dori o nouă formă de autentificare sau un nou tip de utilizator, va trebui să modificăm logica dintr-un singur modul.

În continuare vom prezenta responsabilitățile fiecărui modul în parte și interacțiunile de care e nevoie între acesta și celelalte module pentru implementarea cazurilor de utilizare specificate anterior.

### 4.2.1. Modulul de gestionare a utilizatorilor

După cum îi spune și numele acest modul se ocupă cu gestionarea utilizatorilor aplicației. În cazul de față gestionare înseamnă validare, stocare, accesare și procesare de date despre utilizatori. Principalele responsabilități ale acestui modul sunt:

- Înregistrarea unui utilizator nou
- Autentificarea unui utilizator existent
- Verificarea permisiunilor de acces a unui utilizator existent asupra unei anumite resurse

Acest modul are o interfață directă către utilizator, deci va avea nevoie de pagini și formulare web sugestive, care oferă feedback rapid și sunt ușor de utilizat. De exemplu pentru validarea datelor, în cazurile posibile, pentru un feedback rapid către utilizator, validarea ar trebui făcută instantaneu în front-end.

Acest modul este de sine stătător și poate funcționa fără interacțiunea cu alte module. De menționat că pentru implementarea funcționalităților discutate, alte module au nevoie de a interacționa cu modulu de gestionare a utilizatorilor.

### 4.2.2. Modulul de gestionare a plantelor

Acest modul se ocupă de două mari domenii: cel de gestionarea a tipurilor de plante și cel de gestionare a plantelor ca exemplare. Motivul pentru care am păstrat ambele domenii într-un singur modul este că nu am întrevăzut un tip de schimbare asupra funcționalităților existente în cadrul tipurilor de plante care să nu aducă după ea nevoie unei modificări în cadrul exemplarelor de plante și vice versa.

Modulul de gestionare a plantelor este format din funcționalitățile de

- Vizualizare a tipurilor de plante existente
- Adăugarea unui nou tip de plantă
- Validarea deplinătății informațiilor tipului de plantă

- Vizualizarea exemplarelor de plante din grădina unui utilizator
- Adăugarea unui nou exemplar în grădină
- Vizualizarea informațiilor despre un exemplar din grădină (a nu se confunda cu vizualizarea datelor colectate de echipamentul hardware)

Și acest modul are o interfață directă către utilizator, deci va avea nevoie de pagini și formulare web sugestive, care oferă feedback rapid și sunt ușor de utilizat. De asemenea, acest modul va avea nevoie de un design atrăcțiv, acesta fiind modulul cu care vor interacționa cel mai des majoritatea utilizatorilor.

Acet modul interacționează cu (și depinde de) modulul de gestionare a utilizatorilor pentru:

- Identificarea utilizatorului curent în cazul afișării informațiilor despre grădina acestuia sau a exemplarelor din ea
- Identificarea rolului utilizatorului curent în cazul adăugării de noi tipuri de plante în lista globală sau confirmarea comenziilor

### 4.2.3. Modulul de grădinărit semi-automat

Modulul de grădinărit semi-automat reprezintă partea hardware principală a sistemului. El este responsabil pentru

- monitorizarea mediului în care este planta
- transmiterea valorilor recepționate
- recepționarea valorilor de creștere optimă
- acționarea unor dispozitive capabile să altereze starea mediului

Pentru monitorizarea mediului am ales senzori care să măsoare elementele esențiale creșterii adecvate unei plante. Cele mai importante care intră în această categorie sunt cantitatea de apă de care dispune planta și lumina la care este expusă. Acestea sunt urmate de temperatura ambientală. (Deși temperatura este de asemenea un factor primordial în cardul dezvoltării unei plante, având în vedere că proiectul se adresează plantelor de apartament, unde temperaturile sunt în general adecvate plantelor, am considerat că în cazul nostru, temperatura este un element secundar) După apă, lumină, temperatură, am decis să integrăm și umiditatea aerului.

În Figura 4.7 se poate observa legătura dintre părțile modulului. Aparatele care măsoară date despre mediu trimit informația la logica de control, aceasta, verificând setările pentru specimenul de plantă, compară valorile înregistrate cu cele configurate pentru creșterea propice, iar dacă valorile înregistrate nu sunt în parametrii adecvați, logica de control e responsabilă să acționeze sistemul de control, ca mai apoi acesta să schimbe starea mediului.

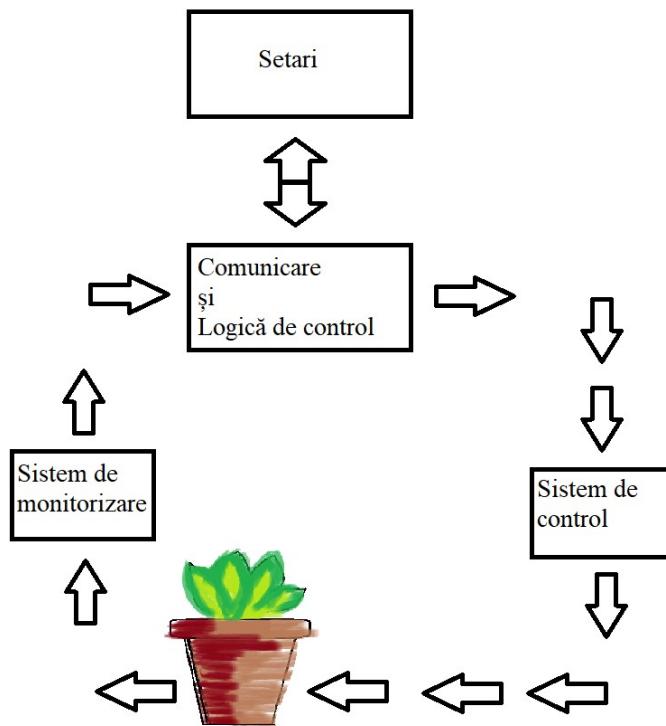


Figura 4.7 Modulul de grădinărit semi-automat

Având în vedere măsurările pe care le luăm din mediu, putem alege care dintre caracteristicile mediului ar trebui să fie alterabile de către sistemul semi-automat și care de către utilizator. Vom observa fiecare posibilitate în parte.

Cantitatea de apă de care dispune o plantă reprezintă, în majoritatea cazurilor, motivul pentru care plantele de apartament își încheie viață. Udatul plantelor reprezintă cea mai costisitoarea activitate din punct de vedere al timpului în întreținerea unei plante. În majoritatea cazurilor este singura activitate care trebuie făcută pe o perioadă de șase luni până la un an.

Deoarece deseori proprietarii uită să își ude plantele, automatizarea acestui element constituie un avantaj substanțial pentru sistemul de grădinărit.

Lumina de care dispune planta este și ea crucială unei dezvoltări propice. În cazul plantelor florale, este nevoie de cât mai multă lumină, deoarece acestea fără o cantitate suficientă de lumină, nu ating inflorescență. Nu este de neglijat nici cazul în care plantele au prea multă lumină, în special lumină directă. De exemplu, planta de aloe vera se arde (arsuri solare) într-o lumină prea puternică, aceasta trebuind să fie expusă doar unei lumini difuze. Pentru a putea controla cantitatea de lumină a unei plante în mod automat putem să utilizăm două sisteme diferite de control: unul care intensifică lumina în cazul în care aceasta nu este suficientă și unul care diminuează lumina în cazul în care aceasta este prea abundantă. În acest scop putem utiliza o sursă de lumină UV și un paravan cu un grad de opacitate redus.

Având în vedere că scopul plantelor de apartament este unul decorativ, și că utilizarea paravanelor său al sursei de lumină ar altera imaginea plantei, am ales să nu

automatizăm acest proces. Un alt motiv al acestei alegeri este că pentru a altera condițiile de luminositate a unei plante tot ce trebuie să facem e să îi schimbăm amplasamentul (să o punem într-o zonă mai luminoasă sau mai întunecată). Aceasta este o acțiune ce trebuie făcută o singură dată, spre deosebire de udarea plantei, astfel că nu o considerăm o acțiune la fel de incomodă pentru utilizator.

Valoarea inadecvată a temperaturii ambientale a plantelor este și aceasta una dintre cauzele morții lor premature. Cel mai întâlnit exemplu este când proprietarii își depozitează plantele iarna deasupra sau în apropierea surselor de căldură precum radiatoarele sau sobele. Pentru a putea controla temperatura ambientală a plantei, este necesară încastrarea plantei într-un container și implementarea unui sistem capabil atât să încălzească containerul respectiv (când temperatura ambientală e prea mică), cât și să îl răcească (când temperatura ambientală este prea mare). Deoarece această funcționalitate este costisitoare atât din punct de vedere estetic dar mai important energetic, și pentru că, la fel ca în cazul cantității de lumină, poate fi reglată cu ușurință de utilizator, am decis să nu o automatizăm.

Un alt aspect asupra căruia punem problema automatizării este umiditatea aerului din jurul plantei. Acest aspect nu este primordial în creșterea plantei și din această cauză, împreuna cu faptul că planta poate fi poziționată lângă aparate electrice am ales să nu îl automatizăm.

În concluzie modulul de grădinărit semiautomat va monitoriza temperatura ambientală, intensitatea luminii, umiditatea solului și a aerului dar va controla doar cantitatea de apă necesară unei plante.

## Capitolul 5. Proiectare de Detaliu si Implementare

Pentru prezentarea din punct de vedere tehnic al aplicației, spre o mai bună reprezentare, vom împărți acest capitol în mai multe secțiuni care vor acoperi, fiecare, specificații despre: aplicația server, aplicația client, modulul hardware, baza de date și legăturile între acestea. O arhitectură conceptuală a sistemului de grădinărit Grădinarul de ghiveci se poate observa în Figura 5.1.

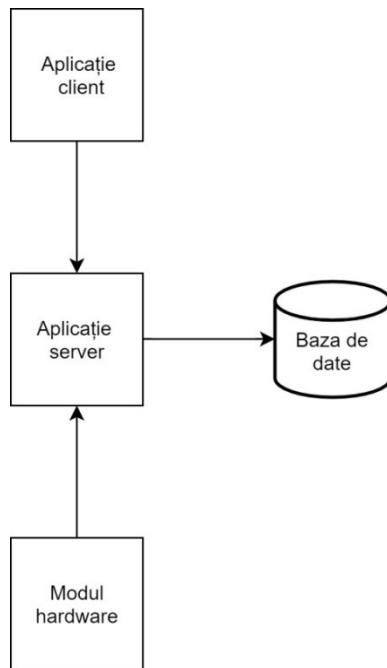


Figura 5.1 Arhitectură conceptuală a Grădinarului de ghiveci

Identificăm că aplicația client comunică doar cu aplicația server, și că nu are nicio legătură directă cu modulul hardware. Reciproca este valabilă: și modulul hardware comunică exclusiv cu aplicația server. Aplicația server răspunde mesajelor de la client și/sau modulul hardware și inițiază comunicare doar cu baza de date.

Prin acest mod de segregare a componentelor am încercat să atingem o decuplare cât puternică a lor, astfel, fiecare componentă putând fi dezvoltată în mod cât mai independent de celelalte.

În urma procesului de design, am ajuns la o arhitectură mult mai complexă ce este vizibilă în Figura 5.2. Aici am reprezentat vizual sistemul dezvoltat cu fundal verde.

Observăm interacțiunile între componentele sistemului reprezentate de săgeți unidirectionale. Faptul că săgeata e unidirecțională nu înseamnă că flow-ul de date este doar într-o direcție. Dimpotrivă, majoritatea interacțiunilor sunt de tipul cerere-răspuns.

Am folosit săgețile unidirectionale pentru evidențierea rolului de client și de server în fiecare comunicare. O săgeată unidirecțională e orientată de la componenta care inițiază comunicare.

Se observă pe fundal alb și limbajele utilizate, framework-urile folosite, tool-urile integrate și aplicațiile externe cu care comunică sistemul nostru. În subcapitolele următoare vom detalia acest proces de design dar și procesul de implementare. Cele mai importante aspecte pe care le vom aborda vor fi alegerile făcute în materie de:

- arhitectura (inclusiv în interiorul modulelor)
- limbaje de programare
- framwork-uri
- diverse unelte și aplicații auxiliare

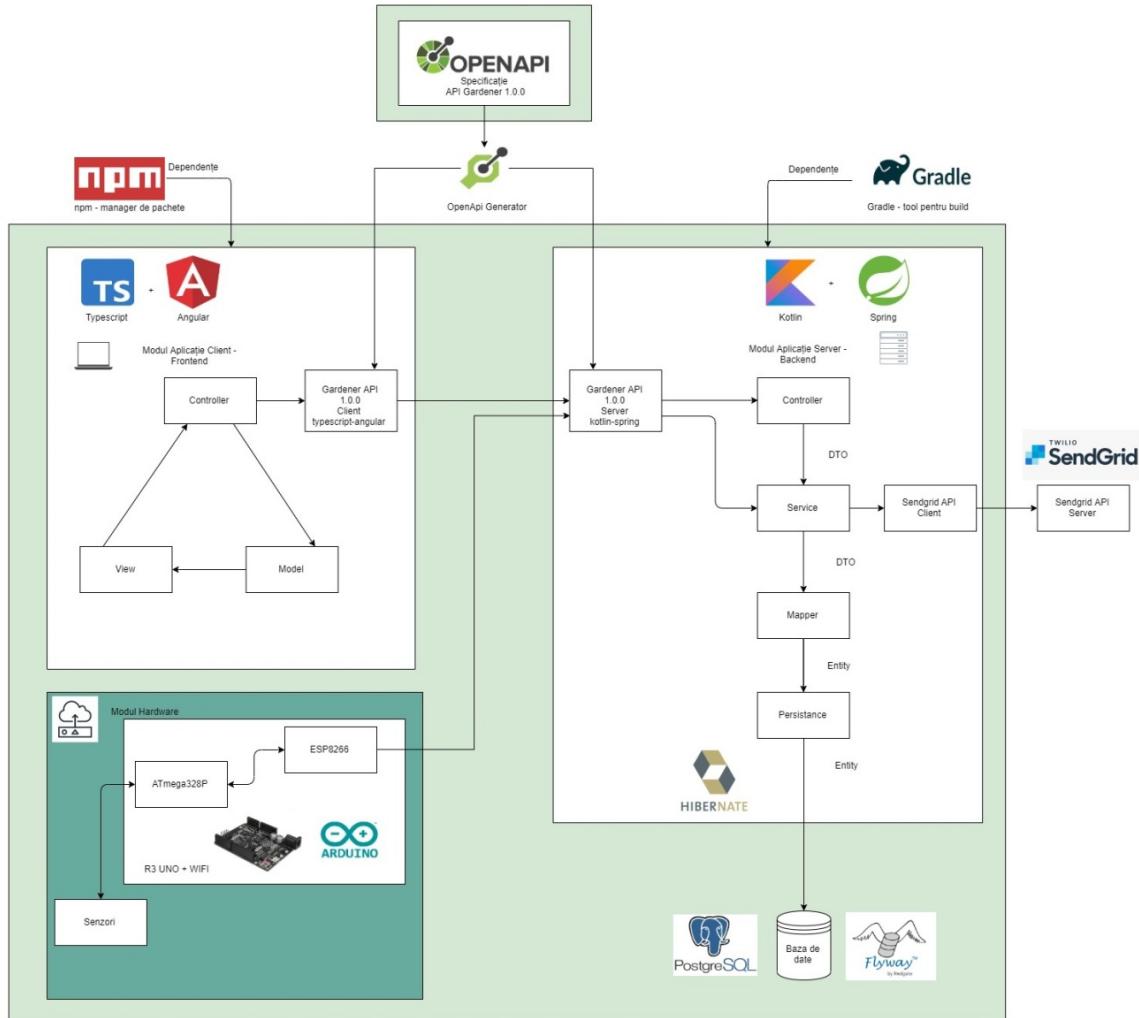


Figura 5.2 Arhitectura detaliată a sistemului

## 5.1. Aplicația server

Aplicația server este centrul logic principal al sistemului. Aceasta e responsabilă de manipularea datelor. Este componenta cu care interacționează celelalte module.

Arhitectura acestei aplicații este „Layered architecture”. Arhitectura pe nivele este formată din:

- Prezentare – Presentation layer
- Logică – Business layer
- Persistența datelor – Persistence layer
- Baza de date – Database layer

**Presentation layer** – reprezintă interfața grafică a unui sistem. E partea cu care utilizatorul interacționează direct. În sistemul de față layer-ul prezentare este reprezentat de aplicația client așa că în aplicația server nu vom avea acest layer. Ce vom avea în schimb este o interfață prin care aplicația client să poată interacționa cu aplicația server. Acest layer este reprezentat de API (Application Programming Interface).

**Business layer** – reprezintă parte de procesare și alterare a datelor. Conform arhitecturii layered în care un layer interacționează doar cu layerurile direct adiacente, acest layer va interacționa cu API-ul și cu layer-ul de persistență.

**Persistence layer** – este responsabil cu interacțiunea directă cu baza de date. Singurul celălalt layer care ar trebui să interacționeze cu acesta este cel de business.

**Database layer** – este reprezentat de baza de sistemul de stocare a datelor. Aceasta poate fi o bază de date sau chiar sistemul de fișier. Pentru aplicația noastră am ales o bază de date externă aplicație server. Aceasta este prezentată în capitolul Figura 5.3.

În concluzie, în cazul nostru, aplicația server va consta din următoarele layere:

- API layer
- Business layer
- Persistence layer

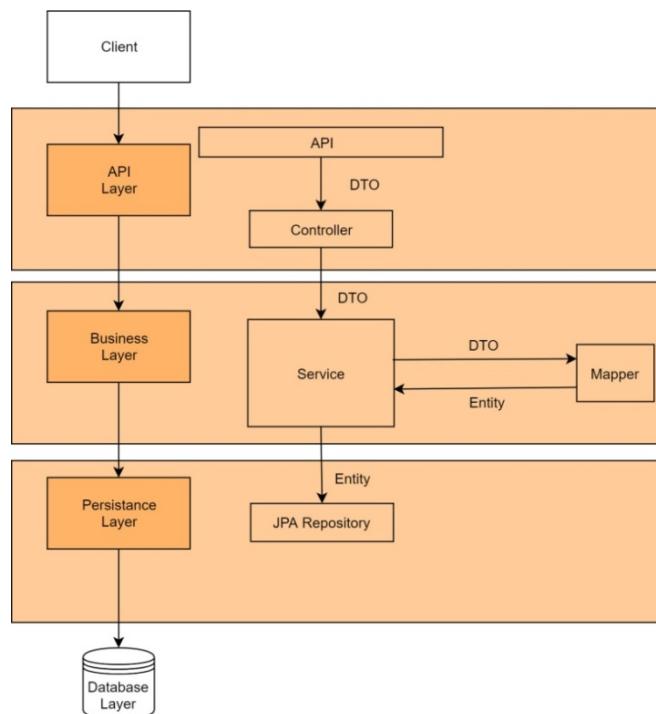


Figura 5.3 Arhitectura aplicației server

În figura Figura 5.3 observăm o arhitectură sistemul nostru în care părțile portocalii aparțin aplicației server.

Conform acestei diagrame și tiparului arhitectural pe nivele, în aplicația noastră toate interacțiunile unui nivel sunt apele spre nivelul direct inferior sau răspunsurile asociate apelelor din nivelul direct superior. Am evidențiat aceasta printr-o diagramă de secvență a cazului de utilizare „Utilizatorul salvează un nou specimen” în Figura 5.4. Dinou, cu portocaliu sunt componentele ce țin de aplicația server.

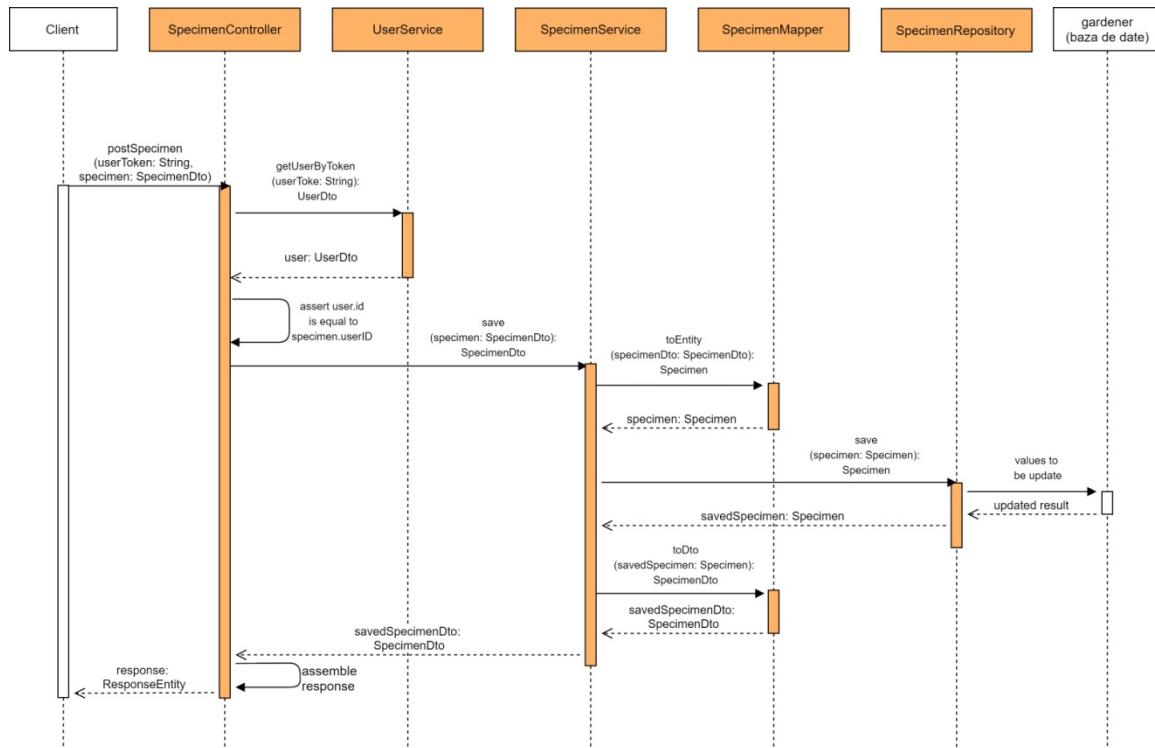


Figura 5.4 Diagramă de secvență pentru cazul de utilizare „Utilizatorul salvează un nou specimen”

Pentru implementarea acestei aplicații am folosit diverse tehnologii cu diferite scopuri. În continuare vom prezenta tehnologiile folosite, vom evidenția motivele pentru care le-am ales și vom prezenta trăsături deosebit de interesante ale acestora unde este cazul.

### 5.1.1. Kotlin

Kotlin este un limbaj de programare bazat pe Java, pentru JVM și Android, care combina funcțiile de programare orientate pe obiecte și funcționale. Se axează pe interoperabilitate, siguranță, claritate și suport pentru unele de dezvoltare.<sup>22</sup> Având în vedere complexitatea medie pe care trebuie să o susțină aplicația server, și multitudinea de limbi de programare pe care le puteam alege, există o gamă largă de opțiuni.

<sup>22</sup> <https://kotlinlang.org>

Kotlin vine și cu mai multe funcționalități ce îl fac atractiv pentru dezvoltatorii software:

- Reduce cantitatea de cod repetitiv („boilerplate code”) pe care dezvoltatorii trebuie să îl scrie
- Oferă tipuri „nullable” pentru a verifica tipul variabilelor la compilare și a evita excepții de tipul „NullPointerException”
- Este un limbaj interoperabil care are compatibilitate 100% cu librăriile JVM dar tot odată poate fi folosit și împreună cu librării de JavaScript

Alegerea Kotlin a fost motivată și de popularitatea în creștere a limbajului, de curiozitate și de ușurința raportată în utilizarea lui. De asemenea de compatibilitatea acestuia cu restul tehnologiilor.

### 5.1.2. *Gradle*

Gradle este un build tool care poate fi folosit pentru diverse libaje de programare. El ajută în controlul procesului de dezvoltare software prin task-uri care se ocupă cu compilarea, împachetarea, testarea, deploy-ul și publicarea aplicațiilor. Instrucțiunile Gradle se scriu într-un limbaj specific Gradle. În locul fișierelor XML, Gradle vine cu propriul DSL (Domain Specific Language) care este mai ușor de citit și scris.<sup>23</sup>

Gradle dispune și de o multitudine de tutoriale și resurse pentru o mai bună înțelegere a acestui tool. Toate acestea sunt gratuite și au fost de ajutor în implementarea proiectului de față<sup>24</sup>.

Alegerea acestui tool e motivată de testarea ușurinței cu care susține că poate fi folosit. De asemenea de compatibilitatea cu restul tehnologiilor. Îl vom utiliza pentru management-ul dependențelor, compilare, împachetare, generare de cod și testare.

### 5.1.3. *Spring boot*

Spring boot este un framework open-source bazat pe Java, folosit în special în dezvoltarea microserviciilor. El oferă o platformă foarte bună pentru dezvoltarea de aplicații „stand-alone”. Se poate utiliza cu un minimum de configurații. Este ușor de înțeles și de folosit deoarece renunță la utilizarea fișierelor XML pentru configurare și se bazează pe adnotările claselor.<sup>25</sup>

Spring oferă și o platformă în care putem configura și genera scheletul unei aplicații de tipul Spring Boot<sup>26</sup>. În figura Figura 5.5 putem observa opțiunile pe care le-am selectat pentru proiectul nostru.

---

<sup>23</sup> <https://gradle.org>

<sup>24</sup> <https://gradle.com/training/>

<sup>25</sup> <https://spring.io/projects/spring-boot>

<sup>26</sup> <https://spring.io/guides/gs/spring-boot/>

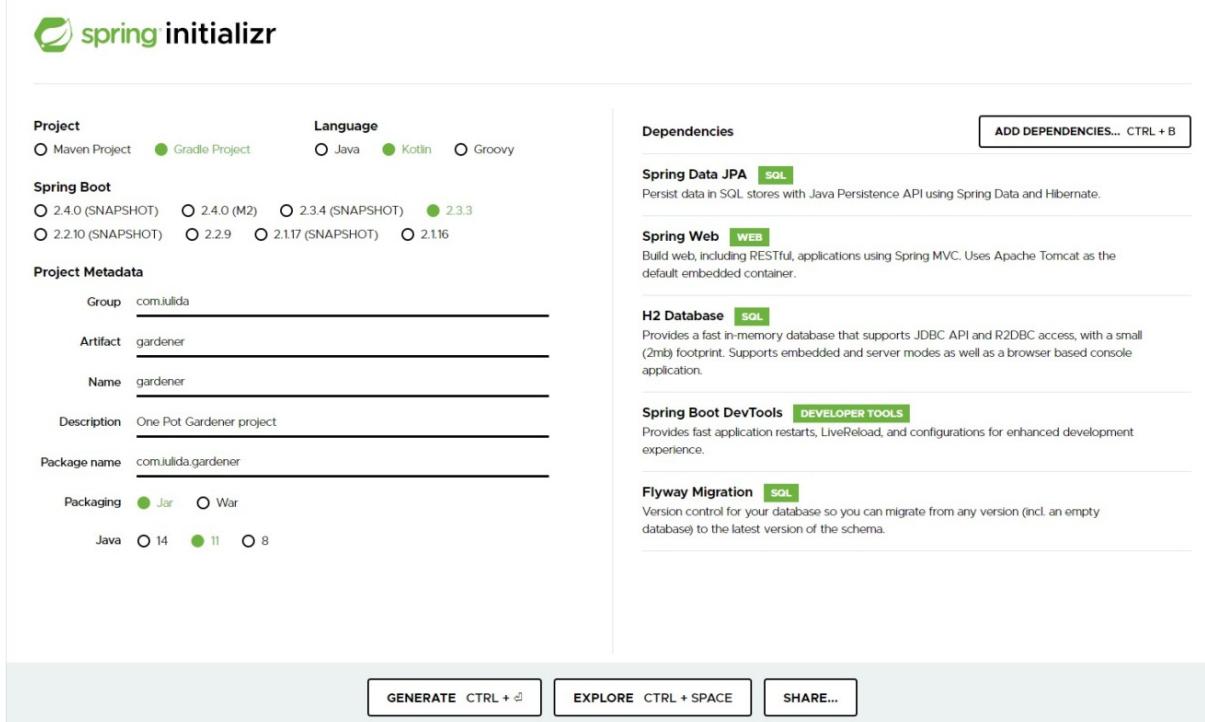


Figura 5.5 Generarea scheletului aplicației server

După cum se observă, cu ajutorul unelei spring initializr<sup>27</sup> am putut crea scheletul unei proiect Gradle scris în Kotlin cu foarte mare ușurință. De asemenea, a fost posibilă adăugarea unor dependințe de bază. Deși a necesitat mai mult studiu incipient, această alegere a fost profitabilă din punct de vedere al timpului de dezvoltare.

#### 5.1.4. OpenAPI

După cum am prezentat la începutul capitolului 5, aplicația server va avea în locul layer-ului de prezentare un layer de API<sup>28</sup>. Un API este o interfață care definește modul de interacțiune dintre două tipuri de actori: server și client. Pentru dezvoltarea API-ului în cadrul aplicației server am apelat la o soluție automată bazată pe standardul OpenAPI Specification<sup>29</sup> și pe tool-ul OpenAPI Generator<sup>30</sup>.

OpenApi Specification este un standard pentru descrierea API-urilor REST (Representational State Transfer) care este ușor de citit și de utilizator și ușor de interpretat de calculator. El este scris în fișiere JSON<sup>31</sup>. În cazul de față am ales să folosim un fișier YAML<sup>32</sup> (YAML Ain't Markup Language) care este un superset al JSON dar în loc de acolade folosește indentările pentru a separa obiectele. Acest lucru îl face mai ușor de citit, motiv principal pentru care l-am ales.

<sup>27</sup> <https://start.spring.io>

<sup>28</sup> <https://en.wikipedia.org/wiki/API>

<sup>29</sup> <https://swagger.io/specification/>

<sup>30</sup> <https://openapi-generator.tech>

<sup>31</sup> <https://www.json.org/json-en.html>

<sup>32</sup> <https://yaml.org>

În figura Figura 5.6 observăm o secțiune din fișierul de specificații OpenAPI al aplicației noastre și cum este acesta interpretat. Pentru interpretarea lui online am utilizat editorul online Swagger Editor<sup>33</sup>.

```

102:
103:     schema: $ref: "#/components/schemas/Error"
104:
105: /plants:
106:   get:
107:     summary: List of all plant objects
108:     operationId: getPlants
109:     tags:
110:       - plants
111:     parameters:
112:       - in: query
113:         name: userToken
114:         required: false
115:         schema:
116:           type: string
117:           description: The token of the user who makes the request
118:     responses:
119:       '200':
120:         description: All plants of the application
121:         content:
122:           application/json:
123:             schema:
124:               $ref: "#/components/schemas/PlantTypeOtos"
125:             default:
126:               description: unexpected error
127:             content:
128:               application/json:
129:                 schema:
130:                   $ref: "#/components/schemas/Error"
131:             summary: Save or update a plant object. If no id is provided, we will
132:             create a new plant. If an id is provided, we will update all the
133:             fields of the plant, this includes updating to 'null' the fields
134:             which are not sent.
135:             operationId: postPlant
136:             tags:
137:               - plants
138:             parameters:
139:               - in: query
140:                 name: userToken
141:                 required: true
142:                 schema:
143:                   type: string
144:                   description: The token of the user who makes the request
145:             requestBody:
146:               description: A plant object
147:               required: true
148:               content:
149:                 application/json:
150:                   schema:
151:                     $ref: "#/components/schemas/PlantTypeOto"
152:             responses:
153:               '201':
154:                 description: created
  
```

**authentication**

- POST** /users/new Create a new user with the given credentials.
- POST** /users/login Login with the given credentials. Retrieve the user token
- POST** /users/logout Logout user with the given token

**plants**

- GET** /plants List of all plant objects
- POST** /plants Save or update a plant object. If no id is provided, we will create a new plant. If an id is provided, we will update all the fields of the plant, this includes updating to 'null' the fields which are not sent.
- GET** /plants/{id} One plant
- DELETE** /plants/{id} Delete the plant with the give id

**features**

- GET** /featureConfigurations List of all feature configuration objects
- POST** /featureConfigurations Save or update a feature configuration object. If no id is provided, we will create a new feature configuration. If an id is provided, we will update all the fields of the feature configuration, this includes updating to 'null' the fields which are not sent.
- GET** /featureConfigurations/{id} One feature configuration
- DELETE** /featureConfigurations/{id} Delete the feature configuration with the give id

**growingConfigs**

Figura 5.6 OpenAPI Specification pentru aplicația server interpretată de Editorul Swagger

Acest fișier de specificații OpenAPI poate fi utilizat pentru a genera atât documentație cum observăm în Figura 5.6 cât și codul reprezentativ. OpenAPI Generator deține o listă impresionantă de generatori atât pentru aplicații server cât și pentru aplicații client. În cazul nostru avem nevoie pentru aplicația server de generarea codului cu generatorul kotlin-spring<sup>34</sup>.

OpenAPI Generator oferă o aplicație CLI<sup>35</sup> dar pentru a automatiza generarea codului API putem să ne folosim de Plugin-ul de Generare OpenAPI de Gradle<sup>36</sup>. Observăm în figura Figura 5.7 că după ce adăugăm acest plugin în Gradle, există niște task-uri în plus. Dintre acestea ne interesează openApiGenerate, pe care îl vom folosi pentru a genera modelele și API-ul aplicație.

```

OpenAPI Tools tasks
-----
openApiGenerate - Generate code via Open API Tools Generator for Open API 2.0 or 3.x specification documents.
openApiGenerators - Lists generators available via Open API Generators.
openApiMeta - Generates a new generator to be consumed via Open API Generator.
openApiValidate - Validates an Open API 2.0 or 3.x specification document.
  
```

Figura 5.7 Task-uri Gradle puse la dispoziție de OpenAPI Tools

<sup>33</sup> <https://editor.swagger.io>

<sup>34</sup> <https://openapi-generator.tech/docs/generators/kotlin-spring>

<sup>35</sup> <https://openapi-generator.tech/docs/installation>

<https://github.com/OpenAPITools/openapi-generator/tree/master/modules/openapi-generator-gradle-plugin>

În figura Figura 5.8 observăm configurarea task-ului de generare a modelelor și serviciilor API.

```
:openApiGenerate { this: OpenApiGeneratorGenerateExtension
    generatorName.set("kotlin-spring")
    inputSpec.set("$rootDir/src/main/resources/api.yaml".toString())
    outputDir.set("$rootDir".toString())
    apiPackage.set("org.openapi.tools.gardener.api")
    modelPackage.set("org.openapi.tools.gardener.model")
    generateModelTests.set(false)
    generateModelDocumentation.set(false)
    generateApiTests.set(false)
    generateApiDocumentation.set(false)

    // https://openapi-generator.tech/docs/generators/kotlin-spring
    configOptions.set(mapOf(
        Pair("dateLibrary", "java8"),
        Pair("interfaceOnly", "true"),
        Pair("library", "spring"),
        Pair("gradleBuildfile", "false"),
        Pair("exceptionHandler", "false"),
        Pair("enumPropertyNaming", "UPPERCASE"),
        Pair("globalProperty", "models,apis")
    ))
}
```

Figura 5.8 Configurarea task-ului openApiGenerate pentru generarea codului de API

Un exemplu al unui endpoint generat este vizibil în figura Figura 5.9. În partea stângă e specificația OpenAPI iar în partea dreaptă e rezultatul generat de plugin.

<pre><code>get:   summary: The statistics registered by for the given gardener/specimen   operationId: getStats   tags:   - greenhouse   parameters:   - in: query     name: userToken     required: true     schema:       type: string       description: The token of the user who makes the request   - in: query     name: gardenerId     required: false     schema:       type: string       format: uuid       description: Filters the results by the given gardenerId   - in: query     name: specimenId     required: false     schema:       type: string       format: uuid       description: Filters the results by the given specimenId   responses:   '200':     description: List of stats for the given params     content:       application/json:         schema:           \$ref: "#/components/schemas/GreenhouseStatsDtos"</code></pre>	<pre><code>@RequestMapping(     value = ["/greenhouse/stats"],     produces = ["application/json"],     method = [RequestMethod.GET]) fun getStats(@NotNull @RequestParam(value = "userToken", required = true) userToken: kotlin.String,             @RequestParam(value = "gardenerId", required = false) gardenerId: java.util.UUID?,             @RequestParam(value = "specimenId", required = false) specimenId: java.util.UUID?):     ResponseEntity&lt;List&lt;GreenhouseStatsDto&gt;&gt; {     return ResponseEntity(HttpStatus.NOT_IMPLEMENTED) }</code></pre>
---	--

Figura 5.9 Specificație OpenApi (stânga) și cod kotlin-spring generat (dreapta)

Deși la o primă vedere e mai mult de scris în fișierul de specificație, e de reamintit că acest fișier poate fi folosit pentru generare de cod și pentru aplicații client. Aceasta este intenția noastră pentru aplicația client a sistemului de grădinărit. Aceasta este prezentată în detaliu în capitolul 5.3.2.

### 5.1.5. Sendgrid

Pentru informarea utilizatorului când planta are nevoie de acțiuni din partea acestuia am apelat la Sendgrid<sup>37</sup> și la API-ul oferit de aceștia deoarece oferă o integrare rapidă și ușoară

### 5.1.6. Spring Data JPA

Pentru implementarea layer-ului de persistență am utilizat Spring Data JPA<sup>38</sup>. Aceata este un modul din Spring Data care se ocupă cu suportul pentru accesarea bazei de date prin specificațiile JPA (Java Persistence API)<sup>39</sup>. Utilizarea lui reduce foarte mult din codul repetitiv de umplutură (boilerplate code) deoarece oferă suport pentru query-uri după valorile field-urilor cât și paginarea rezultatelor.

În figura Figura 5.10 putem observa cum utilizând Spring Data JPA un query pe baza de date devine o singură apelare de metodă.

```
@Repository
interface GreenhouseStatsRepository : JpaRepository<GreenhouseStats, UUID> {

    fun findAllByGardenerId(gardenerId: UUID): List<GreenhouseStats>

    fun findAllBySpecimenId(specimenId: UUID): List<GreenhouseStats>

    fun findAllByGardenerIdAndSpecimenId(gardenerId: UUID, specimenId: UUID): List<GreenhouseStats>

    fun findAllByUserId(userId: UUID): List<GreenhouseStats>
}
```

Figura 5.10 Beneficiile Spring Data JPA evidențiate prin claritatea codului

## 5.2. Baza de date

### 5.2.1. PostgreSQL

Pentru persistarea datelor despre plante, a măsurătorilor despre acestea, a datelor despre Grădinarii de ghiveci, a datelor despre utilizatori și relațiile dintre aceste entități am ales să folosim o bază de date relațională. Pentru implementarea soluției am folosit baza de date relațională open-source PostgreSQL<sup>40</sup>. PostgreSQL este o bază de date open-source care a început ca proiectul POSTGRES la Universitatea Berkeley din California iar acum, cu o vechime de peste 30 de ani, vin în ajutorul dezvoltatorilor cu funcționalități care stochează și scalează și cele mai complicate tipuri de date.

---

<sup>37</sup> <https://sendgrid.com>

<sup>38</sup> <https://spring.io/projects/spring-data-jpa>

<sup>39</sup> <https://www.oracle.com/java/technologies/persistence-jsp.html>

<sup>40</sup> <https://www.postgresql.org>

Pentru vizualizarea datelor am folosit aplicația DBVisualizer<sup>41</sup> cu ajutorul căreia am generat o diagramă ERM (Entity Relationship Diagram) a bazei de date. Aceasta se poate observa în Figura 5.11.

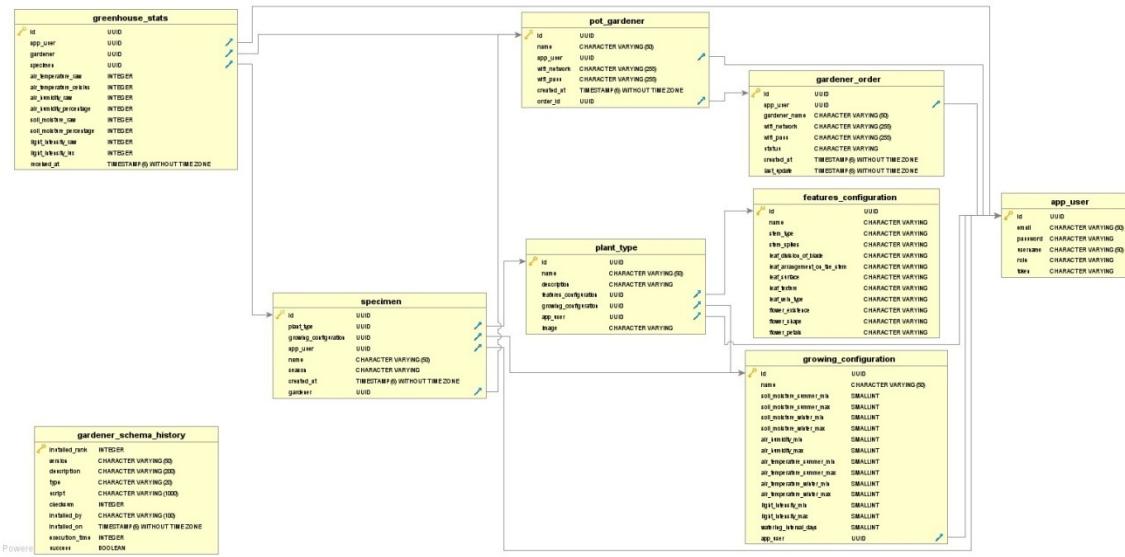


Figura 5.11 Diagrama relațională a entităților din baza de date

Se observă un tabel „gardener\_schema\_history”. Acesta nu conține date despre aplicația în sine ci despre migrările care au adus baza de date în stadiul actual. Aceste migrări sunt practic pașii parcursi pentru ca baza de date să conțină toate tabelele și datele pe care le conține. Baza de date respectă a treia formă normală dar nu respectă respectă forma normală Boyce-Codd. Motivul pentru aceasta este că am ales să stocăm în baza de date, în același tabel, atât valorile citite de la senzor cât și interpretarea lor în unități de măsură adecvate. Am luat această decizie pentru a nu recalculta de fiecare dată valorile afișate (de exemplu procentul de umiditate a solului) din valorile înregistrate (rezistența senzorului de umiditate a solului).

### 5.2.2. Flyway

Pentru construirea bazei de date am folosit sistemul de versionare a bazelor de date Flyway<sup>42</sup>. Scopul utilizării unui sistem de versionare a bazelor de date este să asigure consistență bazei de date în orice mediu de dezvoltare sau execuție. El este compatibil cu Spring și ușor de configurat în fișierul application.properties. În Figura 5.12 sunt vizibile configurațiile pe care le-am făcut pentru baza noastră de date.

```
spring.flyway.url = ${JDBC_DATABASE_URL:jdbc:postgresql://${POSTGRES_HOST:localhost}:${POSTGRES_PORT:5432}/${POSTGRES_DB:gardener}?password=password}
spring.flyway.table = gardener_schema_history
spring.flyway.locations=classpath:flyway/migration
```

Figura 5.12 Configurarea Flyway

<sup>41</sup> <https://www.dbvis.com>

<sup>42</sup> <https://flywaydb.org>

Flyway ne permite să scriem în fișiere separate cod sql care va fi rulat pe baza de date. O astfel de acțiune se numește rularea unei migrări. Flyway păstrează un tabel cu o istorie a tuturor migrărilor executate pe baza de date. Numele standard al acestui tabel e „flyway\_schema\_history”, în cazul nostru l-am numit „gardener\_schema\_history” (vezi Figura 5.11). Cu ajutorul acestui tabel, Flyway știe când o migrare nouă a fost adăugată și nu a fost executată și o să încece să o execute. Când o migrare este executată cu succes, această informație e salvată în „schema\_history”. În Figura 5.13 se pot observa primele migrări pe care le-am executat în sistemul nostru.

insta	versio	description	type	script	checksum	installed_by	installed_on	execution_time	success
[PK]	chara	character varying (200)	charc	character varying (1000)	integer	character varying (100)	timestamp without time zone	integer	boolean
1	1 1	features configuration	SQL	V1__features_configuration.sql	1676362555	postgres	2020-09-04 19:21:47.089906	10	true
2	2 2	gorwing configuration	SQL	V2__gorwing_configuration.sql	1527240976	postgres	2020-09-04 19:21:47.108141	5	true
3	3 3	plant type	SQL	V3__plant_type.sql	-447822976	postgres	2020-09-04 19:21:47.119438	9	true
4	4 4	user	SQL	V4__user.sql	-1878027186	postgres	2020-09-04 19:21:47.135024	13	true
5	5 5	specimen	SQL	V5__specimen.sql	-670010194	postgres	2020-09-04 19:21:47.153657	8	true
6	6 6	gardener order	SQL	V6__gardener_order.sql	1886271602	postgres	2020-09-04 19:21:47.166707	8	true
7	7 7	pot gardener	SQL	V7__pot_gardener.sql	-2071316568	postgres	2020-09-04 19:21:47.179101	9	true
8	8 8	greenhouse stats	SQL	V8__greenhouse_stats.sql	910376092	postgres	2020-09-04 19:21:47.192796	6	true

Figura 5.13 Migrări executate pe baza de date a sistemului

De asemenea, Flyway execută și o verificare a integrității migrărilor deja rulate. El verifică checksum-ul fișierelor actuale cu cel salvat când migrarea a fost executată cu succes. Dacă ele diferă, Flyway aruncă erori de consistență. Astfel, pentru fiecare modificare a bazei de date vom adăuga un nou fișier cu cod SQL ce va fi migrat. În funcție de numele fișierului Flyway știe în ce ordine trebuie executate migrările. Standardul e prefixul „V” urmat de numărul migrării și „\_” (de două ori caracterul underscore) ex: V4\_\_user.sql.

Astfel, utilizând Flyway suntem asigurați de consistența bazei de date și în același timp constrânsi să o respectăm în continuare.

### 5.3. Aplicația client

Aplicația client este responsabilă să ofere o interfață grafică utilizatorului și să comunice cu aplicație server. Ea este implementată ca o aplicație web și este dezvoltată în framework-ul Angular 10.

O aplicație web este o aplicație software care rulează pe un server web. Am ales această opțiune deoarece, spre deosebire de aplicațiile desktop/mobile, care sunt stocate pe un dispozitiv, aceasta este accesibilă de pe orice dispozitiv (desktop sau mobile) cu o conexiune la internet. Angular este un framework de JavaScript open-source pentru aplicațiile web prezentate ca o singura pagina dinamica (single page applications)<sup>43</sup>.

Angular este bazat pe TypeScript<sup>44</sup>. În plus față de JavaScript, TypeScript, după cum îi spune numele asigură consistența tipului datelor, oferind un cadru mai ușor de dezvoltare și mai sigur. Angular vine cu mai multe notiuni utile cum ar fi cea de „componentă”, cea de „double binding” prin care putem conecta o variabilă dintr-un fișier TypeScript cu valoarea unui input din dom.

<sup>43</sup> <https://angular.io>

<sup>44</sup> <https://www.typescriptlang.org>

Arhitectura pe care este bazat framework-ul Angular este MVC (Model View Controller). După cum se observă și în Figura 5.14, aceasta înseamnă că modelul este modificat de către controller în funcție de modificările utilizatorului din nivelul view. De asemenea, view-ul este modificat de către controller în funcție de starea modelului.

De menționat este și angular-cli, aplicația cu interfață în linia de comandă a Angular prin care putem genera, printr-o singură linie comandă o componentă nouă cu toate fișierele ei respective.

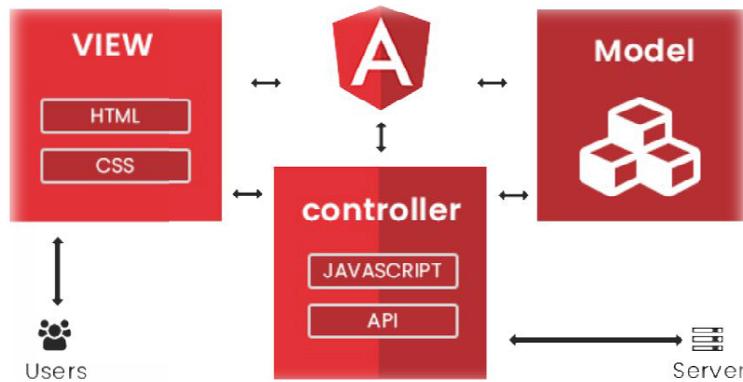


Figura 5.14 Arhitectura MVC în Angular<sup>45</sup>

O trăsătură cheie a Angular sunt componentele. Un component este unitatea de bază pe care este bazată dezvoltarea unei aplicații Angular. Un component poate să agregheze mai multe alte componente. Practic, o aplicație Angular este un arbore de componente.

### 5.3.1. Componentele aplicației

În Figura 5.15 putem observa structura componentelor din aplicația noastră de grădinărit. Am folosit codarea prin culori

- Componentele cu violet fundal sunt vizibile oricărui utilizator care accesează aplicația web.
- Componentele cu fundal verde sunt disponibile doar dacă utilizatorul este autentificat în aplicație.
- Componentele cu fundal roșu sunt disponibile doar utilizatorului „Administrator” când acesta este autentificat în aplicație.

Observăm că există o componentă rădăcină `<gar-root-app>`. Practic, aceasta este componenta din fișierul `index.html`, care este de fapt singurul fișier din aplicație care va fi randat. Componenta `<gar-root-app>` va afișa întotdeauna componente: `<gar-header>` și `<gar-footer>` împreună cu conținutul componentei `<router-outlet>`.

Componenta `<router-outlet>` este specifică Angular. Practic, această componentă nu are un conținut în sine ci va afișa conținutul decis de modulul de routing al Angular. Deciziile acestea se fac în funcție de specificațiile de tip cheie – valoare din componenta

<sup>45</sup> <https://wuschools.com/what-is-mvc-and-understanding-the-mvc-pattern-in-angular/>

de routing. Aici fiecare rută indică ce component ar trebui randat de <router-outlet>. Într-o aplicație Angular putem aveam mai multe astfel de <router-outlet> în funcție de cum creăm structura din modulul de routing. În cazul proiectului de față am folosit o singură astfel de componentă.

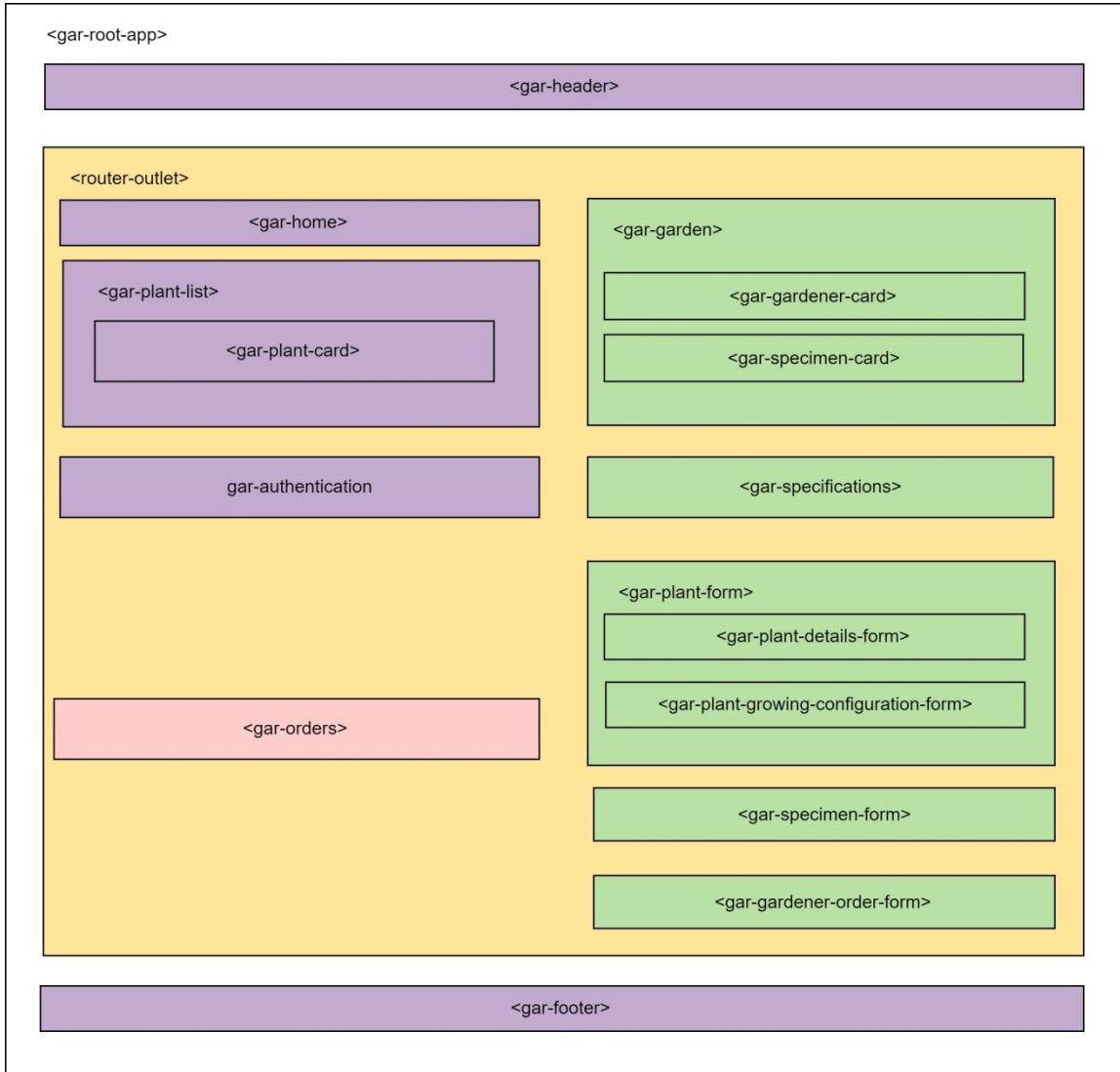


Figura 5.15 Componentele aplicației client

### 5.3.2. OpenAPI, npm, Verdaccio

Pentru generarea API-ului pentru modulul client ne-am folosit de asemenea de fișierul YAML ce conține specificațiile API-ului nostru în formatul OpenAPI Specification și de OpenAPI Generator. De această dată am folosit generatorul typescript-angular<sup>46</sup> pentru a genera un client API. Folosind managerul de pachete npm<sup>47</sup> am instalat dependențele acestui client și am creat un pachet de npm cu API-ul nostru.

<sup>46</sup> <https://openapi-generator.tech/docs/generators/typescript-angular>

În Figura 5.16 se poate observa în stânga aceeași specificație OpenAPI pe care am folosit-o în aplicația server iar în dreapta, de această dată este codul generat pentru aplicația client.

```

/*
 * Gardener API
 * No description provided (generated by Openapi Generator https://github.com/openapitools/openapi-generator)
 *
 * The version of the OpenAPI document: 1.0.0
 *
 * NOTE: This class is auto generated by OpenAPI Generator (https://openapi-generator.tech).
 * https://openapi-generator.tech
 * Do not edit the class manually.
 */
import ...

export declare class GreenhouseService {
    protected httpClient: HttpClient;
    protected basePath: string;
    defaultHeaders: HttpHeaders;
    configuration: Configuration;
    encoder: HttpParameterCodec;
    constructor(httpClient: HttpClient, basePath: string, configuration: Configuration);
    private addToHttpParams;
    private addToHttpParamsRecursive;
    /**
     * The statistics registered by for the given gardener/specimen
     * @param userToken The token of the user who makes the request
     * @param gardenerId Filters the results by the given gardenerId
     * @param specimenId Filters the results by the given specimenId
     * @param observe set whether or not to return the data Observable as the body, response or events. defaults to returning the body.
     * @param reportProgress flag to report request and response progress.
     */
    getStats(userToken: string, gardenerId?: string, specimenId?: string, observe?: 'body', reportProgress?: boolean, options?: {
        httpHeaderAccept?: 'application/json';
    }): Observable<Array<GreenhouseStatsDto>>;
    getStats(userToken: string, gardenerId?: string, specimenId?: string, observe?: 'response', reportProgress?: boolean, options?: {
        httpHeaderAccept?: 'application/json';
    }): Observable<HttpResponse<Array<GreenhouseStatsDto>>>;
    getStats(userToken: string, gardenerId?: string, specimenId?: string, observe?: 'events', reportProgress?: boolean, options?: {
        httpHeaderAccept?: 'application/json';
    }): Observable<HttpEvent<Array<GreenhouseStatsDto>>>;
}

getStats(userToken: string, gardenerId?: string, specimenId?: string, observe?: 'body', reportProgress?: boolean, options?: {
    httpHeaderAccept?: 'application/json';
}): Observable<Array<GreenhouseStatsDto>>;
getStats(userToken: string, gardenerId?: string, specimenId?: string, observe?: 'response', reportProgress?: boolean, options?: {
    httpHeaderAccept?: 'application/json';
}): Observable<HttpResponse<Array<GreenhouseStatsDto>>>;
getStats(userToken: string, gardenerId?: string, specimenId?: string, observe?: 'events', reportProgress?: boolean, options?: {
    httpHeaderAccept?: 'application/json';
}): Observable<HttpEvent<Array<GreenhouseStatsDto>>>;

```

Figura 5.16 Specificație OpenAPI (stânga) și codul generat de generatorul typescript-angular (dreapta)

Pentru utilizarea locală a acestui pachet am ales să îl publicăm într-un registry local utilizând aplicația Verdaccio<sup>48</sup>. Verdaccio oferă posibilitatea publicării pachetelor npm într-un registry privat, local prin rularea unui server pe portul 4873 al localhost care se comportă ca un registry de npm.

Pentru instalarea acestui pachet am modificat fișierul .nprc al aplicație Angular astfel încât atunci când npm instalează dependențele aplicației, pentru pachetele din modulul nostru să acceseze registry-ul local, nu cel oficial de la npm. În Figura 5.17 se pot observa modificările aduse fișierului .npmrc.

```

registry=https://registry.npmjs.org
@com.iulia.local:registry=http://localhost:4873

```

Figura 5.17 Modificarea fișierului .npmrc pentru utilizarea registry-ului local oferit de Verdaccio

## 5.4. Modulul Hardware

Modulul hardware reprezintă partea din sistemul de grădinărit care se ocupă în principal cu monitoarizarea plantei și controlarea mediului acesteia pentru o dezvoltare propice dar și cu informarea utilizatorului cu privire la starea mediului. Pentru ca acest modul să își atingă scopul de a furniza date utilizatorului (dar și de a menține umiditatea solului în parametrii specificați de utilizator) el trebuie să comunice cu aplicația server. În

<sup>47</sup> <https://www.npmjs.com/package/npm>

<sup>48</sup> <https://verdaccio.org>

Figura 5.18 observăm comunicarea bidirectională dintre aplicația server și modulul hardware. Modulul hardware furnizează aplicației server valorile citite de la senzori. Aplicația server transmite valoarea minimă pentru umiditatea solului.

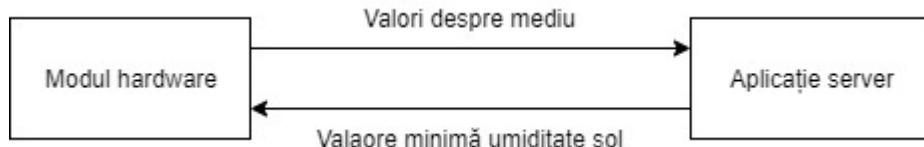


Figura 5.18 Comunicarea între modulul hardware și aplicația sever

Conform concluziei capitolului 3, pentru sistemul de control din modulul hardware utilizăm o placă R3 Uno + WiFi. Pentru dezvoltarea codului necesar acestui modul am folosit limbajul de programarea Arduino și mediu de dezvoltare Arduino Software<sup>49</sup>. Înem să amintim arhitectura plăcuței R3 Uno WiFi este compusă din modulul ATmega328P și modulul ESP8266. Aceste module sunt programabile independent dar pot comunica între ele. Plăcuța R3 UNO + WiFi poate funcționa 5 moduri distincte. Ele sunt evidențiate în Figura 5.19. Pentru a selecta unul dintre acele moduri, trebuie introdusă configurația respectivă pe cele 8 switch-uri ale plăcuței. Modul de programare ATmega328P, de exemplu este selectat prin setarea switch-urilor în pozițiile (OFF, OFF, ON, ON, OFF, OFF, OFF, OFF).

	1	2	3	4	5	6	7	8
CH340 connect to ESP8266 (upload sketch)	OFF	OFF	OFF	OFF	ON	ON	ON	NoUSE
CH340 connect to ESP8266 (connect)	OFF	OFF	OFF	OFF	ON	ON	OFF	NoUSE
CH340 connect to ATmega328 (upload sketch)	OFF	OFF	ON	ON	OFF	OFF	OFF	NoUSE
Mega328+ESP8266	ON	ON	OFF	OFF	OFF	OFF	OFF	NoUSE
All modules work independent	OFF	NoUSE						

Figura 5.19 Moduri funcționare plăcuță R3 Uno + WiFi

Vom împărți modulul nostru hardware în două submodule mai mici în funcție de responsabilitățile acestora:

- Primul este „Modulul de grădinărit”. Acesta este responsabil pentru citirea datelor de la senzori și acționarea pompei.
- Al doilea este „Modulul de comunicare”. El este format din modulul ESP8266 și este responsabil cu comunicarea cu Modulul de grădinărit și cu aplicația server.

În Figura 5.20 se observă cum am împărțit aceste responsabilități și cum comunică modulele între ele.

<sup>49</sup> <https://www.arduino.cc/en/main/software>

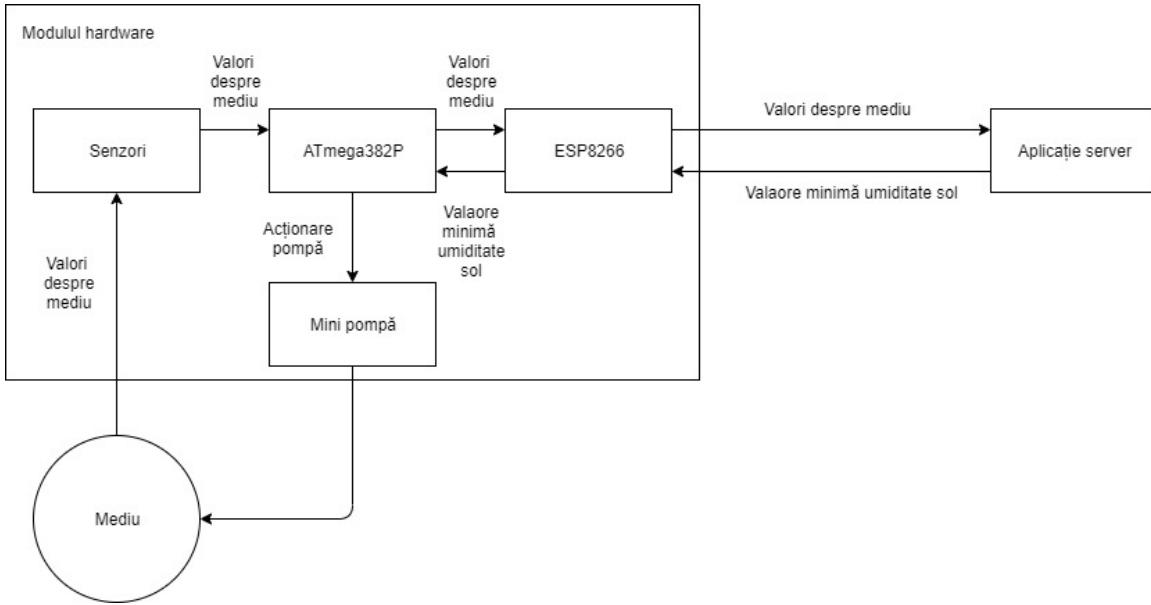


Figura 5.20 Modul hardware – Responsabilități

De menționat că în Figura 5.20 modulul hardware poate fi privit din cele trei componente principale pe care le-am mai prezentat:

- Elemente de monitorizare a mediului – în cazul nostru senzorii
- Logica de control – reprezentată de placă R3 UNO + WiFi, care la rândul ei este compusă din
  - Microcontrollerul ATmega382P și
  - Modulul ESP8266
- Elemente de alterare a mediului – în cazul nostru mini pompa de apă

În continuare, vom prezenta în două subcapitole fiecare dintre responsabilitățile și implementarea celor două submodule. Un al trei-lea subcapitol va prezenta modul acestor de comunicare între ele.

#### 5.4.1. Submodulul de grădinărit – ATmega382P

După cum am prezentat în subcapitolul 2.1.8, datele de care suntem interesați să le cunoaștem despre mediu sunt: temperatura aerului, umiditatea aerului, temperatura solului și intensitatea luminoasă a mediului. Pentru a putea măsura aceste valori am intrebuințat senzori compatibili cu R3 Uno + WiFi.

Pentru a măsura temperatura și umiditatea aerului am ales să folosim senzorul DHT11. Acesta este un senzor cu funcție dublă, care poate măsura atât temperatura aerului, cât și umiditatea din el. Am ales acest senzor și din considerente financiare. În Figura 5.21 se poate observa senzorul DHT11. Acest senzor poate măsura temperaturi între 0 și 50 de grade Celsius. Umiditatea aerului este măsurată de acest senzor în valori de la 0 la 100, unde 100 înseamnă ploaie<sup>50</sup>.

<sup>50</sup> <https://www.adafruit.com/product/386>

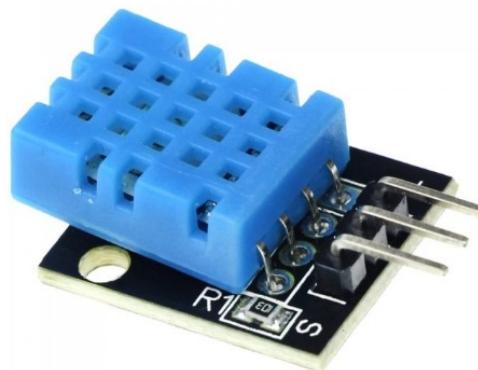


Figura 5.21 Senzor DHT11<sup>51</sup>

Pentru a măsura umiditatea solului am ales un modul compus din senzorul FC28 și cipul LM393. Acestea pot fi observate în Figura 5.22. Cele două plăcuțe expuse ale senzorului analogic funcționează ca sonde, acționând ca un rezistor variabil. Cu cât este mai multă apă în sol cu atât conductivitatea dintre plăcuțe va fi mai mare și rezistența mai mică. Senzorul înregistrează valori între 0 și 1023<sup>52</sup>.

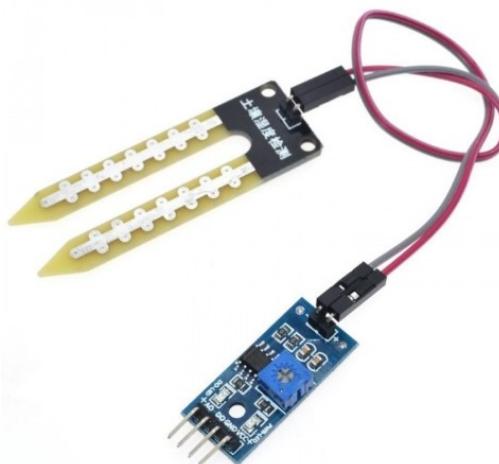


Figura 5.22 Modul cu senzor de umiditate sol FC28 și cip LM393<sup>53</sup>

Măsurarea intensității luminoase a mediului o vom face folosind un fotorezistor și o rezistență fixă. Un fotorezistor (LDR - Light Decreasing Resistance) este un element fotoconductor. Aceasta înseamnă că această componentă electronică își crește conductivitatea (pierde din rezistență) cu cât este expus la o lumină mai puternică. În

<sup>51</sup> <https://cleste.ro/senzor-temperatura-si-umiditate-dht11.html>

<sup>52</sup> <https://www.circuitstoday.com/arduino-soil-moisture-sensor>

<sup>53</sup> <https://cleste.ro/modul-cu-senzor-umiditate-sol.html>

Figura 5.23 observăm fotorezistorul folosit. Pentru a putea măsura cu cât fotorezistorul devine mai mult sau mai puțin conductiv, folosim o rezistență fixă pentru a crea un divizor rezistiv.<sup>54</sup>



Figura 5.23 Fotorezistor<sup>55</sup>

Ulterior citrii datelor din mediu, vom modifica mediul în funcție de configurația plantei. După cum am prezentat în capitolul 4, singurul element pe care îl vom modifica este umiditatea solului. Pentru aceasta am ales să utilizăm o mini pompă submersibilă. Deoarece am dorit alimentarea mini-pompei de la o altă sursă de tensiune, și nu din plăcuță, am utilizat și un releu în combinație cu aceasta. În Figura 5.24 observăm în stânga mini pompa submersibilă iar în dreapta un modul de releu de 5V.

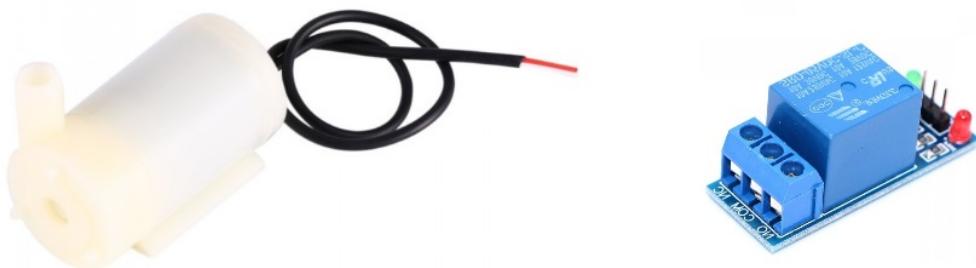


Figura 5.24 Mini pompa submersibilă<sup>56</sup> și Modul releu de 5V<sup>57</sup>

---

<sup>54</sup> <https://en.wikipedia.org/wiki/Photoresistor>

<sup>55</sup> <https://cleste.ro/fotorezistor-5528-ldr.html>

<sup>56</sup> <https://cleste.ro/pompa-de-apa-3-6v.html>

<sup>57</sup> <https://cleste.ro/modul-releu-1-canal-5v.html>

După cum se observă și în Figura 5.19 pentru a citi informațiile furnizate de acești senzori și pentru controla pompa de apă avem nevoie de modul „CH340 connect to ATmega328” al plăcuței. Acesta este modul în care plăcuța se comportă ca un Arduino Uno.

Pentru a obține date de la senzori, îi conectăm la pinii analogi ai cipului ATmega328P, setăm acești pini ca pini de intrare și citim valorile prin funcția specifică Arduino „analogRead”.

Pentru a controla mini pompa de apă, conectăm intrarea acesteia la un pin digital al ATmega328P și scriem pe acesta valori „HIGH” sau „LOW”, cu funcția specifică Arduino „digitalWrite”.

Din cauza sensibilității senzorului de temperatură și umiditate DHT11 avem nevoie să folosim și timpi morți în cod pentru a nu citi valori eronate.

### 5.4.2. Submodulul de comunicare – ESP8266

Funcția de comunicare, atât cu aplicația server cât și cu modulul de grădinărit îi revine modului de comunicare. Pentru implementarea acestui modul vom scrie cod, de asemenea, în Arduino Software dar de data aceasta vom încărca codul pe modulul ESP8266. Pentru a aduce plăcuța în starea în care codul poate fi încărcat pe ESP8266, aceasta trebuie să fie în starea „CH340 connect to ESP8266 (upload sketch)”. Pentru aceasta switch-urile plăcuței trebuie să fie în starea (OFF, OFF, OFF, OFF, ON, ON, ON, OFF) conform Figura 5.19.

Modulul de comunicare execută următoare secvență de pași

- Se conectează la rețeaua de WiFi. Credențialele pentru aceasta trebuie specificate în codul sursă. Credențialele se referă la numele rețelei de WiFi și parola acesteia.
- Se conectează la aplicația server. Adresa aplicație server trebuie specificată în codul sursă.
- Așteaptă valorile transmise de modulul de grădinărit
- Citește valorile transmise de modulul de grădinărit
- Face un apel HTTP de tipul POST cu valorile citite către aplicația server
- Așteaptă maximum 5 secunde răspunsul de la aplicația server. În caz că aplicația server nu răspunde în 5 secunde, se reîncearcă conectarea la server.
- Primește răspunsul de la aplicația server. Răspunsul constă în valorile de configurare pentru modulul de grădinărit
- Procesează răspunsul de la aplicația server. De exemplu elimină spațiile din răspuns și caracterul de rând nou
- Trimit răspunsul procesat către modulul de grădinărit
- Începe de la primul pas.

Pentru a putea încărca acest cod pe plăcuță a trebuit să importăm un manager de plăcuță specific ESP8266<sup>58</sup> și să selectăm plăcuța „Generic ESP8266 Module” cu viteza de încărcare de 115200, flash size de 512K și flash mode „DIO”.

---

<sup>58</sup> [http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)

### 5.4.3. Comunicare dintre submodulele modulului hardware

Pentru ca modulul ATmega382P și modulul ESP8266 să poată comunica între ele, plăcuța trebuie să fie în starea „Mega328+ESP8266”, ceea ce înseamnă conform Figura 5.19 că switch-urile acesteia trebuie să fie în starea (ON, ON, OFF, OFF, OFF, OFF, OFF, OFF).

Modulele comunică între ele prin circuite fizice. Fiecare modul are o componentă UART (Asynchronous Receiver/Transmitter) care este capabilă să transmită și să primească date. Componentele UART au două canale de comunicare. Unul pentru transmiterea datelor: Tx, și unul pentru recepționarea datelor: Rx<sup>59</sup>. În Figura 5.25 se observă cum sunt două componente UART conectate între ele pentru a putea comunica. Componentele UART transmit datele în mod asincron.

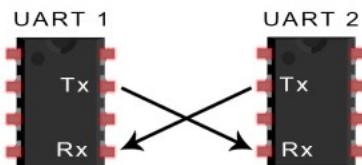


Figura 5.25 Comunicare între două componente UART

Comunicarea între cele două module se face prin funcția Arduino „Serial” care trebuie inițializată la aceeași viteză în ambele module.

Metoda pe care am ales-o pentru a implementa comunicarea dintre submodulul de grădinărit și submodulul de comunicare urmează pașii în buclă infinită:

- Submodulul de comunicare așteaptă un mesaj de la submodulul de grădinărit prin UART
- Submodulul de grădinărit trimite datele citite de la senzori sub formă de JSON prin UART
- Submodulul de grădinărit așteaptă 10 secunde
- Submodulul de comunicare trimite un request de tipul POST către aplicația server
- Submodulul de comunicare primește răspunsul de la aplicația server
- Submodulul de comunicare transmite răspunsul către submodulul de grădinărit prin UART
- Submodulul de grădinărit citește mesajul prin UART

## 5.5. Alte instrumente

În dezvoltarea sistemului de față am utilizat și alte instrumente de dezvoltare software. Deoarece au adus un plus de ușurință în dezvoltare am decis că merită prezentate succint în următoarele capitole.

---

<sup>59</sup> <https://www.circuitbasics.com/basics-uart-communication>

### 5.5.1. Git

În dezvoltarea sistemului de grădinărit am folosit aplicația de versionare git<sup>60</sup>. Ca repository online am folosit platforma oferită de GitHub<sup>61</sup>. Versionarea codului a adus un imens ajutor în dezvoltare și depanare iar faptul că am folosit și un repository online a eliminat necesitatea de a crea back-up-uri manuale ale proiectului. Am utilizat git inclusiv pentru redactarea documentației.

### 5.5.2. IntelliJ IDEA

IntelliJ IDEA este un IDE (Integrated Development Environment)<sup>62</sup> care ne-a permis să creștem productivitatea în dezvoltarea sistemului, în special utilizând funcționalitatea acestuia de „coding assistance” (asistentă în codare) și „autocomplete” (autocompletare). Am folosit acest IDE atât pentru dezvoltarea aplicației Spring Boot în Kotlin cât și pentru dezvoltarea aplicației Angular în Typescript.

### 5.5.3. Arduino Software (IDE)

Pentru scrierea de cod care să ruleze pe plăcuța R3 UNO + WiFi am folosit IDE-ul Arduino Software<sup>63</sup>. În acesta se scriu snippet-uri de cod numite schițe care mai apoi sunt încărcate pe plăcuță. Arduino Software IDE facilitează atât scrierea de cod, cât și configurarea modului de încărcare a schițelor pe plăcuțe.

### 5.5.4. Heroku

Pentru a avea aplicația accesibilă în Cloud am apelat la Heroku<sup>64</sup>. Aceasta este o aplicație de tipul PaaS (Platform as a Service) care oferă posibilitatea să rulăm codul în Cloud fără a ne ocupa de infrastructura ce este în general asociată cu această operație.

---

<sup>60</sup> <https://git-scm.com>

<sup>61</sup> <https://github.com>

<sup>62</sup> <https://www.jetbrains.com/idea/>

<sup>63</sup> <https://www.arduino.cc/en/main/software>

<sup>64</sup> <https://dashboard.heroku.com/apps>

## Capitolul 6. Testare și Validare

Având în vedere că proiectul de față conține atât module software cât și hardware vom aborda testarea din punct de vedere al dezvoltării software și testarea din punct de vedere a proiectării hardware.

Testarea software reprezintă un mecanism care verifică dacă un sistem îndeplinește cerințele specificațiilor sale atât funcționale cât și non-funcționale. Testarea unui sistem se poate împărtăși în mai multe categorii, în funcție de discriminantul pe care îl alegem.

În funcție de modul de execuție al testului, testarea poate fi manuală sau automată. Testarea manuală este efectuată de către un utilizator iar cea automată de către un sistem automat.

Pot cataloga modurile de testare și în funcție de tipul testului. Acesta poate testa o componentă individuală (unit test) sau un întreg flux dintr-un sistem, caz în care vorbim de un test de integrare.

Dacă în cazul dezvoltării software, pentru ca testele să treacă, e nevoie ca programul să fie implementat în prealabil, în cazul proiectării hardware există o suită de teste ce trebuie să treacă și înainte începerii proiectării. Aici ne referim la testarea părților componente ale sistemului ce urmează a fi proiectat.

### 6.1. Testarea manuală

Principalul mod de testarea al aplicației l-a reprezentat în lucrarea de față testarea manuală. Deoarece am ținut spre un sistem care este ușor de folosit de utilizator, a fost nevoie de testare manuală din punct de vedere al experienței utilizatorului (UX testing<sup>65</sup>) astfel că o dată cu testarea acesteia am testat și părțile funcționale ale aplicației.

Întrucât aplicația client și cea server au fost dezvoltate separat și testarea lor s-a realizat în fază incipientă separat.

Fiind că aplicația de frontend nu prezintă o logică complexă ce poate fi testată, testarea manuală a aplicației client înainte de a o conecta pe aceasta la aplicația server s-a rezumat la următoarele

- Paginile sunt randate conform așteptărilor – am testat aceasta prin accesarea paginilor în browser la diferite rezoluții
- Rutele redirecționează utilizatorul spre paginile configurate – am testat aceasta prin accesarea rutelor și observarea noii pagini randate
- Elemente interactive de CSS precum: pagina de home dispune de scroll de tipul smooth

Pentru testarea manuală a aplicației server am utilizat aplicația Postman<sup>66</sup>. Postman este o platformă de dezvoltare a API-urilor ce oferă diverse funcționalități, de la client pentru testarea API-ului la colecții partajate de modele de request-uri, servere pentru mocking sau tool-uri de documentare.

---

<sup>65</sup> <https://userpeek.com/blog/user-experience-testing/>

<sup>66</sup> <https://www.postman.com>

În cazul nostru am utilizat aplicația desktop Postman pentru funcționalitatea acesteia de a acționa ca un client ce poate să trimită requesturi HTTP specifice REST către aplicația server.

În figura Figura 6.1 observăm un apel de tipul GET către aplicația server care rulează la adresa <http://localhost:8080>. Observăm mai jos răspunsul aplicației la acest apel. Inspectând răspunsul vedem că este consistent cu starea bazei de date, deci putem afirma că acest endpoint din API funcționează.

```

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 1024
Date: Mon, 12 Jun 2017 10:44:47 GMT
Connection: keep-alive
ETag: "1593444487"
Last-Modified: Mon, 12 Jun 2017 10:44:47 GMT
X-Powered-By: PHP/7.0.13
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
Content-Security-Policy: default-src 'self'; script-src 'self' https://code.jquery.com; style-src 'self' https://code.jquery.com; font-src 'self' https://code.jquery.com; img-src 'self' https://www.flowerpower.com.au/media/catalog/product/cache/1e4231be311893e047a7d21f6ed/g/gardenia-augusta-magnifica.jpg;
https://www.flowerpower.com.au/media/catalog/product/image/2314392187/gardenia-gold-magic.jpg;
https://www.flowerpower.com.au/media/catalog/product/cache/1e4231be311893e047a7d21f6ed/g/gardenia-augusta-magnifica.jpg;
https://www.flowerpower.com.au/media/catalog/product/image/2314392187/gardenia-gold-magic.jpg";
object-src 'none';
frame-ancestors 'none';

{
    "id": 42,
    "name": "Yellow warning rice",
    "category": "Yellow warning rice",
    "color": "#FFFF00",
    "description": "https://www.australiaplantsonline.com.au/pub/media/catalog/product/cache/1e4231be311893e047a7d21f6ed/g/gardenia-augusta-magnifica.jpg",
    "image": "https://www.flowerpower.com.au/media/catalog/product/image/2314392187/gardenia-gold-magic.jpg",
    "humidity": 99,
    "temperature": 22,
    "lightIntensity": 9999,
    "airTemperature": 15,
    "airHumidity": 99,
    "airTemperatureMin": 18,
    "airTemperatureMax": 22,
    "lightIntensityMin": 9999,
    "lightIntensityMax": 99999,
    "soilMoisture": 44,
    "soilMoistureMin": 89,
    "soilMoistureMax": 22,
    "soilMoistureInter": 55,
    "userId": null,
    "wateringInterval": 5
}
  
```

Figura 6.1 Trimitere apel http GET către server din Postman

În Figura 6.2 observăm trimiterea de date către aplicația server din Postman prin intermediul unu apel http POST cu care reprezintă o nouă statistică înregistrată de un grădinar.

```

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 1024
Date: Mon, 12 Jun 2017 10:44:47 GMT
Connection: keep-alive
ETag: "1593444487"
Last-Modified: Mon, 12 Jun 2017 10:44:47 GMT
X-Powered-By: PHP/7.0.13
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
Content-Security-Policy: default-src 'self'; script-src 'self' https://code.jquery.com; style-src 'self' https://code.jquery.com; font-src 'self' https://code.jquery.com; img-src 'self' https://www.flowerpower.com.au/media/catalog/product/cache/1e4231be311893e047a7d21f6ed/g/gardenia-augusta-magnifica.jpg;
https://www.flowerpower.com.au/media/catalog/product/image/2314392187/gardenia-gold-magic.jpg;
https://www.flowerpower.com.au/media/catalog/product/cache/1e4231be311893e047a7d21f6ed/g/gardenia-augusta-magnifica.jpg;
https://www.flowerpower.com.au/media/catalog/product/image/2314392187/gardenia-gold-magic.jpg";
object-src 'none';
frame-ancestors 'none';

{
    "userId": "e97131e-8259-45ca-885c-986738532e8f",
    "gardenerId": "832fead3-aee4-4aef-9b85-47c443e7637",
    "airTemperature": 99,
    "airHumidity": 99,
    "soilMoisture": 999,
    "lightIntensity": 2999
}
  
```

Figura 6.2 Trimitere apel http POST către server din Postman

Verificând baza de date cu aplicatia pgAdmin<sup>67</sup>, în Figura 6.3 observăm că mesajul a fost întradevăr salvat cu succes. Și acest scenariu de testarea manuală a fost îndeplinit cu succes.

The screenshot shows the pgAdmin interface with the following details:

- Browser:** Shows two servers: 'Herkules' and 'PostgreSQL 12'. Under 'PostgreSQL 12', there are databases ('gardener'), schemas ('public'), and tables ('greenhouse\_stats').
- Query Editor:** Contains the SQL query: `select * from greenhouse_stats order by received_at desc;`
- Data Output:** Displays the results of the query. The table has columns: id [PK], app\_user, gardener\_uuid, specimen\_uuid, air\_temperature\_raw, air\_temperature\_celsius, air\_humidity\_raw, air\_humidity\_percentage, soil\_moisture\_raw, and soil\_moisture\_percent. The results show 256 rows affected.
- Messages:** Shows a green message at the bottom right: "Successfully run. Total query runtime: 80 msec. 256 rows affected."

Figura 6.3 Baza de date vizibilă în aplicația pgAdmin

## 6.2. Testarea automată

Pentru aplicația de server am recurs și la testarea automată. Am folosit în acest scop o configurație Spring pentru testare și o bază de date pentru teste care să fie diferită de cea pe utilizată în modul de dezvoltare. Am folosit baza de date h2<sup>68</sup>. Aceasta este o bază de date de tipul „in memory”. Ea este creată în momentul când aplicația pornește și distrusă când aplicația se oprește. În Figura 6.4 observăm configurările făcute pentru această bază de date. Și pe această bază de date rulăm migrările Flyway, deci orice test pe care o să îl scriem va include testarea integrității migrărilor și execuția acestora.

```

spring.datasource.driver-class-name = org.h2.Driver
spring.datasource.password = ''
spring.datasource.url = jdbc:h2:mem:foo;DB_CLOSE_ON_EXIT=TRUE
spring.datasource.username = sa

spring.flyway.url = jdbc:h2:mem:foo;DB_CLOSE_ON_EXIT=TRUE
spring.flyway.user = sa
spring.flyway.password = ''
spring.flyway.table = gardener_schema_history
spring.flyway.locations = classpath:flyway/migration
spring.flyway.baseline-on-migrate=true

```

Figura 6.4 Configurări bază de date h2

<sup>67</sup> <https://www.pgadmin.org>

<sup>68</sup> <https://www.h2database.com/html/main.html>

Am ales să scriem în testarea automată teste de integrare pentru componenta utilizator. În Figura 6.5 observăm scenariile de test pe care le-am implementat.

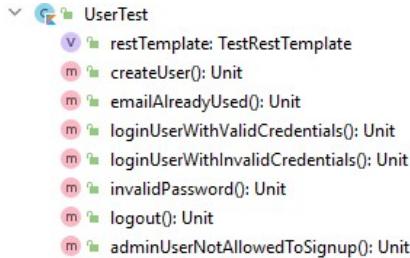


Figura 6.5 Scenarii de test pentru componenta User

Pentru aceste scenrii ne-am folosit de clasa TestRestTemplate a pachetului org.springframework.boot.test.web.client care simplifică foarte mult apelarea API-ului aplicației. În Figura 6.6 observăm scenariul de test pentru logare și delogarea unui utilizator. Prima dată creăm un cont, ne autentificăm, salvăm token-ul primit ca răspuns, folosim token-ul pentru delogare, suntem delogați iar la următoare încercare de delogare verificăm că token-ul respectiv a fost șters de fapt prima dată, acum ne mai fiind găsit.

```

@Test
fun logout() {
    var userDto = UserDto(
        email = "test6@email.com",
        password = "password123",
        username = "test6",
        role = UserRole.USER
    )
    var signUpResponse = restTemplate.postForEntity( url: "/v1/users/signup", userDto, UserDto::class.java)
    assert(signUpResponse.hasBody())
    var loginResponse = restTemplate.postForEntity( url: "/v1/users/login", userDto, UserDto::class.java)
    assert(loginResponse.hasBody())
    assert(loginResponse.body!!.token != null)
    var token = loginResponse.body!!.token
    var logoutResponse = restTemplate.postForEntity( url: "/v1/users/logout?userToken=" + token, request: null, String::class.java)
    assert(logoutResponse.statusCode == HttpStatus.OK)
    var invalidLogoutResponse = restTemplate.postForEntity( url: "/v1/users/logout?userToken=" + token, request: null, String::class.java)
    assert(invalidLogoutResponse.statusCode == HttpStatus.NOT_FOUND)
}

```

Figura 6.6 Test pentru delogarea unui utilizator

### 6.3. Testarea componentelor

Testarea componentelor are ca scop confirmarea faptului că piesele folosite pentru dezvoltarea unui sistem funcționează în parametrii specificați. Pentru partea hardware a sistemului de grădinărit am testat toate piesele componente în mod individual înainte de a începe asamblarea modulului hardware.

#### 6.3.1. Testarea plăcuței R3 UNO + WiFi

Pentru testarea plăcuței R3 UNO + WiFi am validat patru cazuri de test separate:

- Testarea conexiunii plăcuței cu laptopul
- Testarea modulului ATmega328P
- Testarea modulului ESP8266
- Testarea comunicării dintre modulul ATmega328P și ESP8266

Pentru testarea conexiunii plăcuței cu laptopul scenariul pe care l-am urmat a fost ca după conectarea plăcuței la laptop prin cablul micro USB, am căutat că aceasta întrădevăr e recunoscută și apare în dispozitivele periferice vizibile de către Managerul de Dispozitive ale sistemului de operare. În cazul în care aceasta nu apare ca dispozitiv periferic, poate fi o problemă cu plăcuța în sine sau cu cablul micro USB folosit. Există cabluri micro USB doar pentru încărcare, care nu folosesc pinii de transmitere a datelor sau există plăcuțe care sunt defecte din fabricație.

Pentru testarea modulului ATmega328P am avut nevoie să verificăm dacă acesta funcționează independent. Pentru aceasta a fost nevoie să setăm plăcuța în modul de operare „CH340 connect to ATmega328 (upload sketch)” și să selectăm tipul de plăcuță „Arduino Uno” în Arduino Software, după cum se observă și în Figura 6.7.

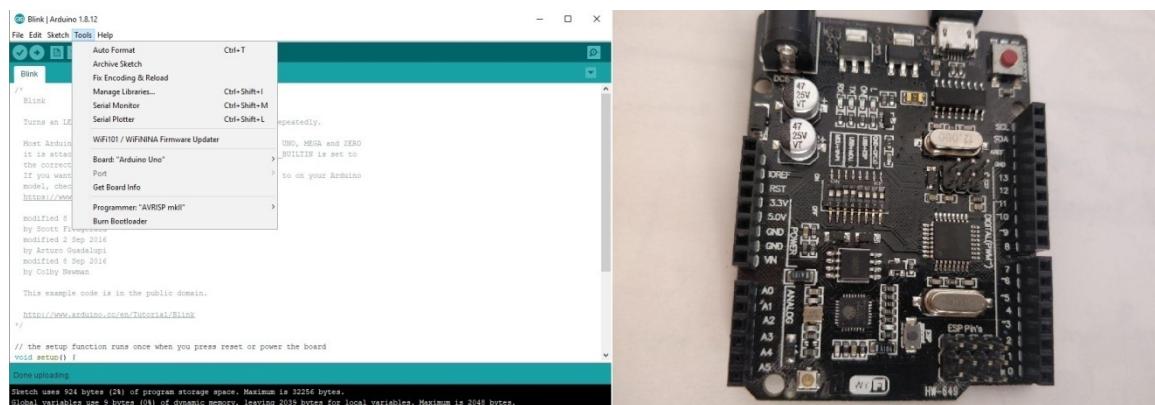


Figura 6.7 Setările pentru utilizare ATmega328P

Apoi, am ales un exemplu de cod sursă din biblioteca Arduino. Exemplul ales a fot „Blink”<sup>69</sup>. Acesta făcea ledul încorporat al plăcuței să se stingă și să se aprindă la un interval regulat. Observăm în imaginea din Figura 6.8 plăcuță în cu ledul în stare stinsă în stânga și cu ledul în stare aprinsă în dreapta.

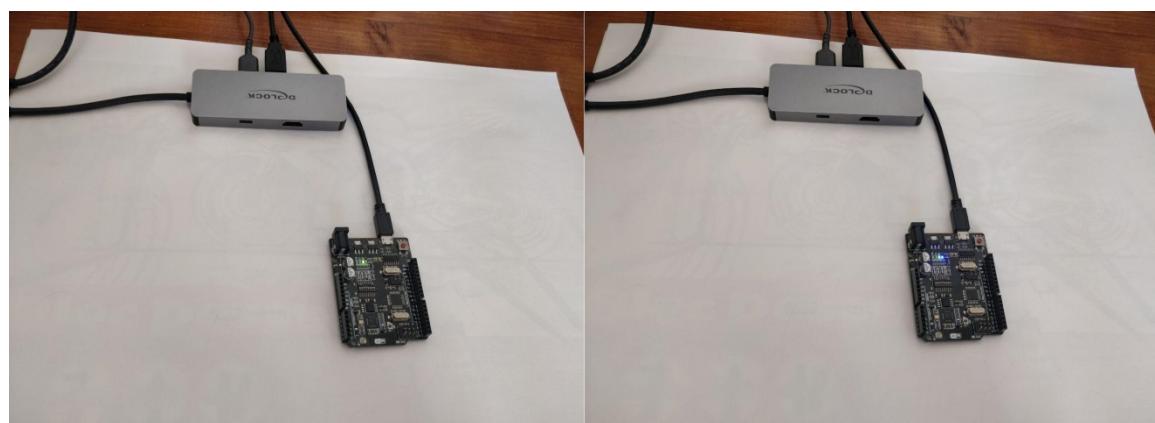


Figura 6.8 Testare ATmega328P

Al treilea scenariu de test făcut în testarea plăcuței a fost testarea modulului ESP8266. Pentru aceasta a fost nevoie să setăm plăcuța în modul de operare „CH340

<sup>69</sup> <https://www.arduino.cc/en/Tutorial/Blink>

connect to ESP8266 (upload sketch)” și să selectăm tipul de plăcuță „Generic ESP8266 Module” în Arduino Software, după cum se observă și în Figura 6.9. Pentru a avea acces la acest tip de plăcuță a trebuit în prealabil să importăm un manager de plăcuță specific ESP8266<sup>70</sup>.

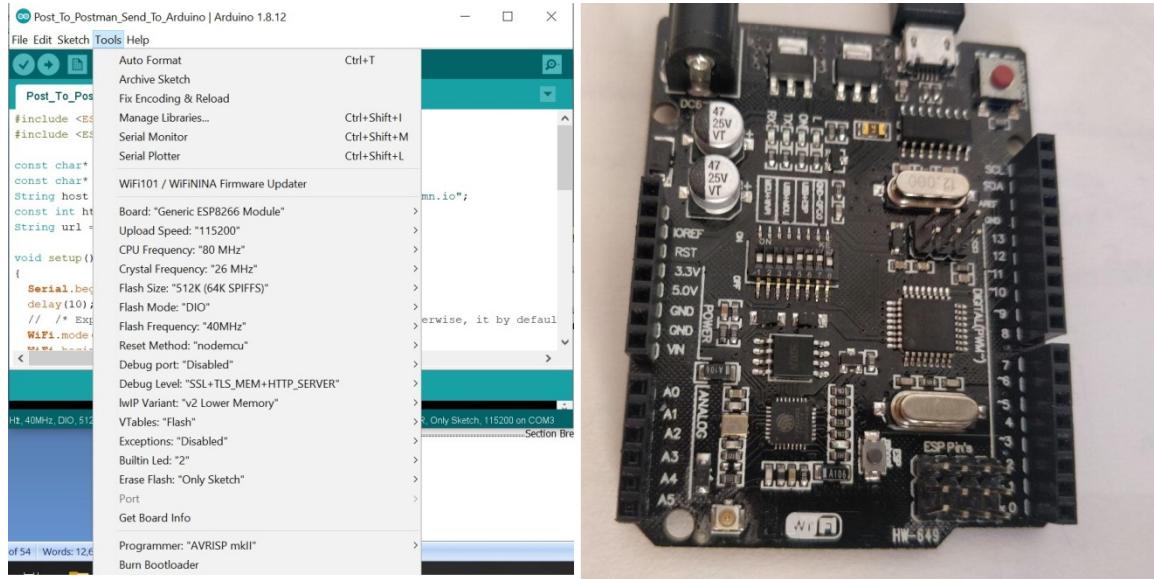


Figura 6.9 Setările pentru utilizare ESP2866

Pentru testarea modulului am folosit același cod din schița anterioară, „Blink” doar că l-am adaptat pentru un led legat la pinul 2 al modulului ESP8266. În figura Figura 6.10 se observă atât starea în care ledul este oprit în stânga, cât și starea în care ledul este aprins, conform codului.

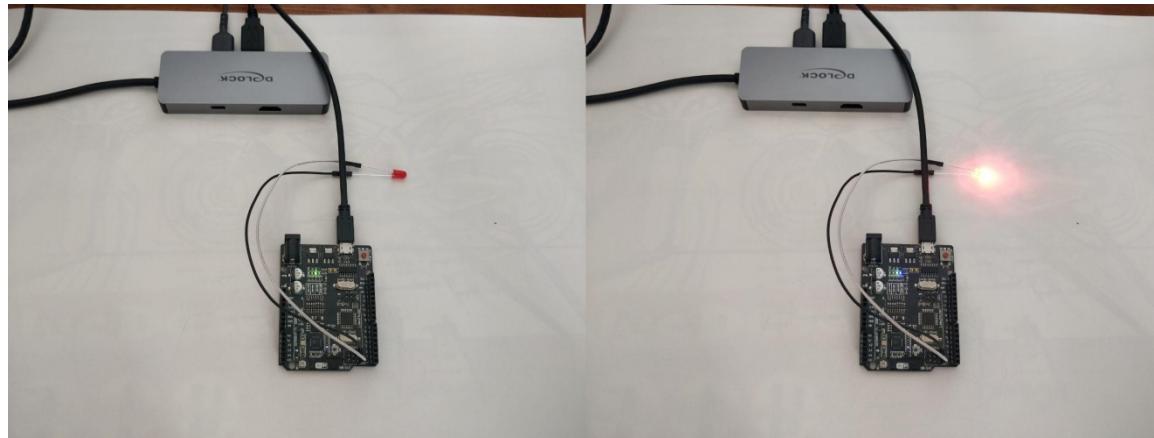


Figura 6.10 Testare ESP8266

<sup>70</sup> [http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)

Pentru acest modul am testat și faptul că se poate conecta la internet. Pentru aceasta am folosit aplicația Postman<sup>71</sup> unde am creat un „mock server” (un serviciu oferit de Postman prin care putem crea un server pe care mai apoi îl configuro să răspundă în funcție de anumite exemple pe care noi le creăm<sup>72</sup>).

Scenariul a fost să trimitem un apel http de tipul POST către serverul creat și să afișăm răspunsul de la acesta. În Figura 6.11 se poate observa atât serverul creat în Postman cât și ce afișează modulul ESP8266 prin comunicarea serială.

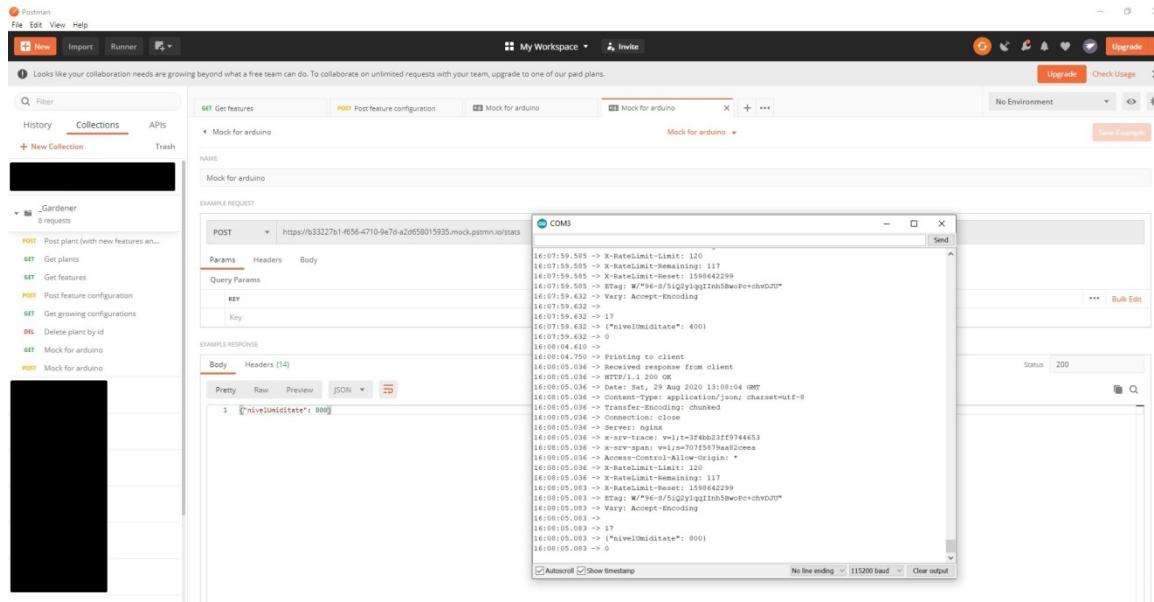


Figura 6.11 Serve mock Postman și Monitorul serial Arduino

Pentru testarea comunicării dintre modulele ATmega328P și ESP8266 am folosit și un senzor de umiditate și temperatură, DHT11. Senzorul a fost testat individual în prealabil. Procesul și rezultatele scenariului de test pentru senzor sunt prezentate în subcapitolul 6.3.2 Testarea senzorului DHT11. Recomandăm parcurgerea aceluia capitol și apoi revenirea la acesta.

Pentru testarea comunicării dintre cele două module am ales următorul scenariu: am conectat senzorul de umiditate și temperatură la un pin analog al ATmega328P și am transmis prin modul de comunicare serial valorile de temperatură citite de senzor, iar în modulul de ESP8266 am citit datele primite prin comunicarea serială, urmând ca atunci când aceste valori depășeau pragul de 30 de grade, să aprindem un led conectat la un pin digital al modului ESP8266.

Pașii urmăți pentru implementarea acestui scenariu au fost:

- Setarea plăcuței în modul „CH340 connect to ATmega328 (upload sketch)”
- Încărcarea codului care citește temperatură de la senzor și o transmite serial

<sup>71</sup> <https://www.postman.com>

<sup>72</sup> <https://learning.postman.com/docs/designing-and-developing-your-api/mocking-data/setting-up-mock/>

- Setarea plăcuței în modul „CH340 connect to ESP8266 (upload sketch)” și resetarea acesteia (apăsarea butonului de reset sau deconectarea urmată de reconectarea plăcuței la laptop)
- Încărcarea codului care citește valorile transmise prin comunicarea serială și aprinde ledul dacă valoarea primită e mai mare de 30
- Setarea plăcuței în modul „Mega328+ESP8266”

Ultimul pas este vizibil în Figura 6.12. Când plăcuța este în acest mod de operare ea rulează simultan codul de pe cele două module și conectează componentele UART ale acestor între ele ca acestea să poată comunica serial.

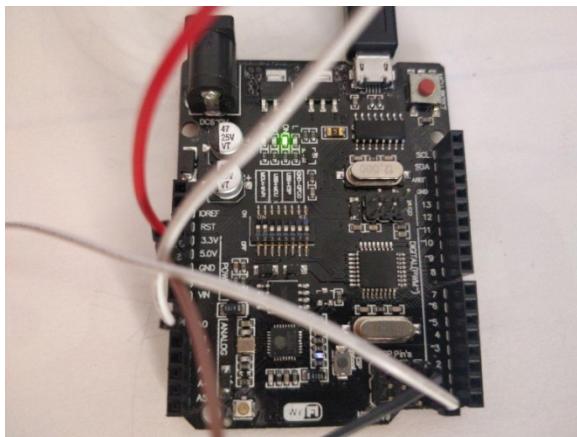


Figura 6.12 Plăcuța setată în modul de operare „Mega328+ESP8266”

Pentru validarea scenariului am crescut temperatura înregistrată de senzorul conectat la ATmega328P cu ajutorul unui feon. După câteva secunde, led conectat la modulul ESP8266 s-a aprins (Figura 6.13 stânga). Pentru a scădea temperatura înregistrată de senzor am pulverizat apă rece pe acesta. După câteva secunde ledul a început să pâlpâie și apoi s-a stins (Figura 6.13 dreapta). Pâlpâirea ledului a însemnat că acesta a înregistrat valori alternativ mai mici și mai mari de 30 de grade pentru câteva secunde, iar stingerea sa că a înregistrat doar valori mai mici de 30 de grade. Fluctuații mici de temperatură sunt de așteptă, astfel că am considerat testul executat cu succes.

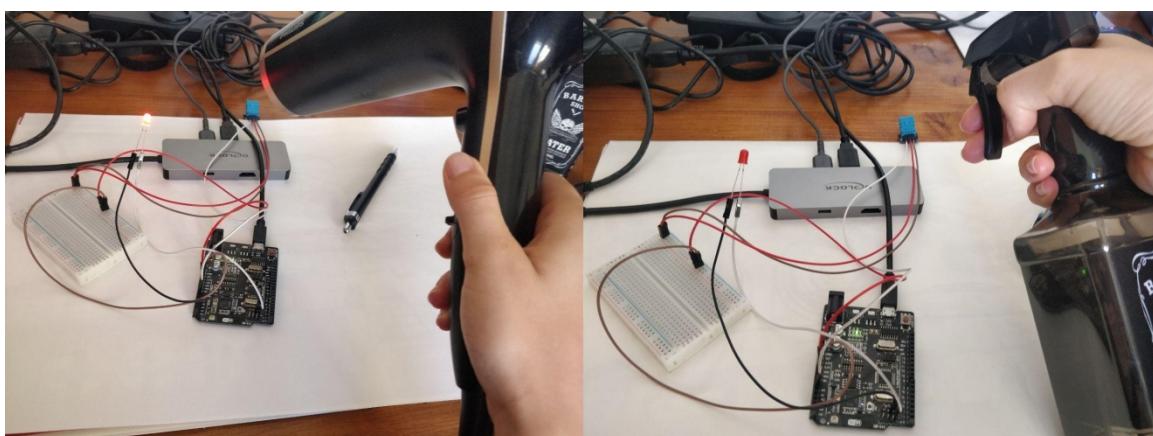


Figura 6.13 Rezultatele testării comunicării dintre modulul Atmeg328P și ESP8266

Conform rezultatelor obținute în urma acestor scenarii de testarea am concluzionat că placă R3 UNO + WiFi funcționează corect și este pregătită să susțină implementarea proiectului.

### 6.3.2. Testarea senzorului DHT11

Senzorul DHT11 (Figura 5.21) este un senzor care măsoară temperatura și umiditatea din aer. Am conectat acest senzor la placă R3 Uno + WiFi conform unui tutorial prezentat de<sup>73</sup> în care este specificat că avem nevoie și de o bibliotecă adițională<sup>74</sup> pentru a putea citi datele de la DHT11.

Scenariul de test pe care l-am folosit a fost să printăm valorile înregistrare de senzor pe canalul serial, să modificăm starea mediului din exterior și să observăm în fereastra „Serial Monitor” a mediului de dezvoltare ce valori sunt înregistrare. De menționat că rata de transfer setată în codul sursă trebuie să fie aceeași cu rata de transfer a ferestrei de comunicare.

Observăm în Figura 6.14 că în momentul în care pulverizăm apă pe senzor, acesta înregistrează o temperatură din ce în ce mai scăzută și o umiditate a aerului din ce în ce mai ridicată.

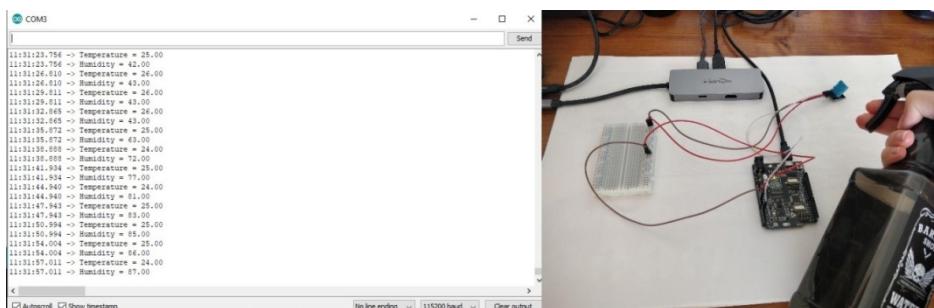


Figura 6.14 Măsurători înregistrare de senzorul DHT11 sub acțiunea umezirii

Pentru a testa și că senzorul își revine din această stare am accelerat procesul prin utilizarea unui feon. În Figura 6.15 se observă cum senzorul înregistrează temperaturi din ce în ce mai ridicate și o umiditate a aerului tot mai scăzută.

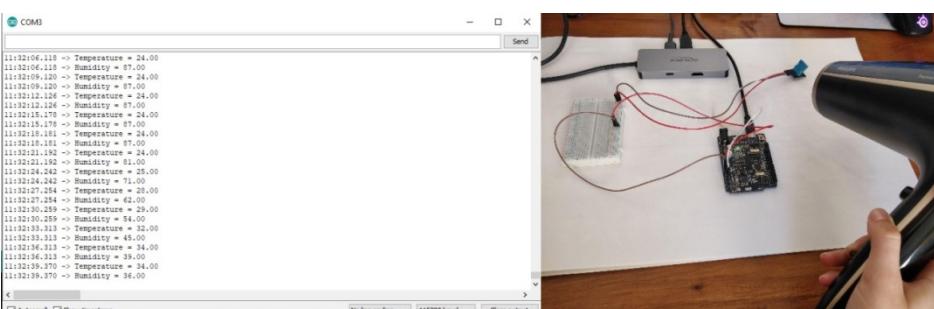


Figura 6.15 Măsurători înregistrarea de senzorul DHT11 sub acțiunea încălzirii

<sup>73</sup> <https://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino/>

<sup>74</sup> <https://www.circuitbasics.com/wp-content/uploads/2015/10/DHTLib.zip>

Conform rezultatelor obținute în urma acestui scenariu de testarea am concluzionat că senzorul DHT11 pentru măsurarea temperaturii și a umidității aerului funcționează în mod adecvat.

### 6.3.3. Testarea modulului de măsurarea a umidității solului

Pentru testarea modulului de măsurare a umidității solului am abordat aceeași metodă ca în subcapitolul 6.3.2. și anume înregistrarea valorilor citite de senzor și afișarea acestora în pe monitorul serial.

În Figura 6.16 dreapta observăm senzorul de umiditate a solului introdus într-un recipient cu un pământ extrem de uscat în care nu ar putea nicio plantă să se dezvolte. Valorile citite de sensor în acest mediu sunt expuse în figura 6.8 stânga. Observăm că doarece senzorul este într-un mediu cu o conductivitate extrem de scăzută, înregistram valori aproape maxime ale rezistenței.

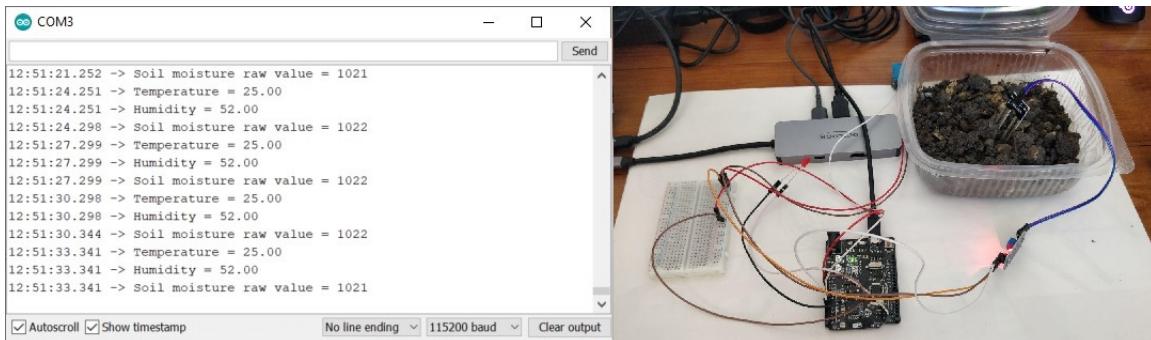


Figura 6.16 Măsurători înregistrate de modulul de umiditate a solului într-un mediu uscat

În Figura 6.17 am introdus senzorul de umiditate în recipientul în care creștem un avocado din sâmbure. Observăm în partea dreaptă a figurii că senzorul este introdus aproape total în apă. În partea stângă a figurii 6.9 sunt prezentate valorile înregistrate de senzor în acest mediu. Fiind un mediu cu o conductivitate mare, modulul înregistrează valori ale rezistenței sub jumătate ca în cazul anterior.

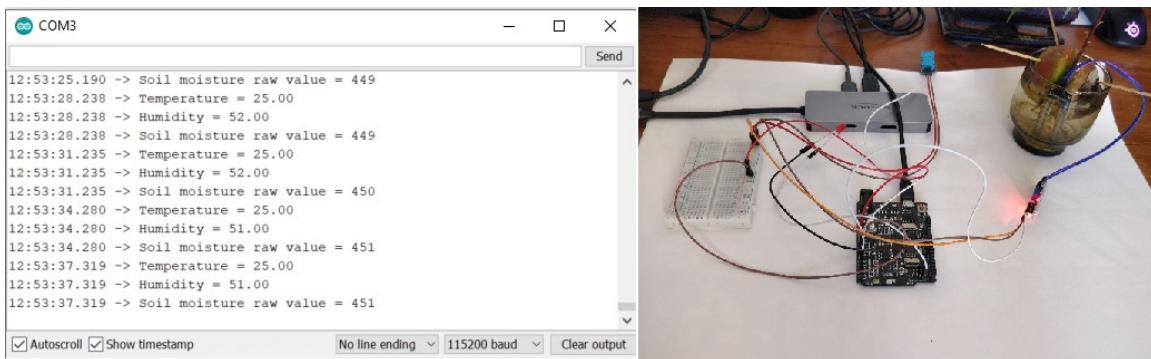


Figura 6.17 Măsurători înregistrare de modulul de umiditate a solului într-un mediu umed

Conform rezultatelor obținute în urma acestui scenariu de testarea am concluzionat că modulul pentru măsurarea umidității solului funcționează în mod adecvat.

#### 6.3.4. Testarea fotorezistorului

Pentru testarea fotorezistorului am folosit și o rezistență fixă iar conectarea acestora la plăcuță R3 UNO + WiFi a fost identică cu modul în care sunt acestea conectate la o plăcuță Arduino UNO în Figura 6.18.

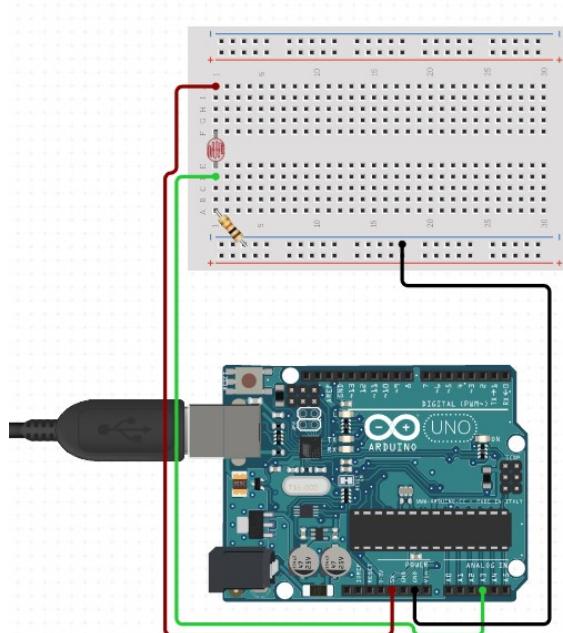


Figura 6.18 Schemă pentru conectarea fotorezistorului și rezistenței

Ca în scenariile anterioare de testare a senzorilor, am transmis datele citite serial și am folosit monitorul serial pentru a le vizualiza. Observăm în Figura 6.19 stânga valorile înregistrate când fotorezistorul este într-o cameră întunecoasă.

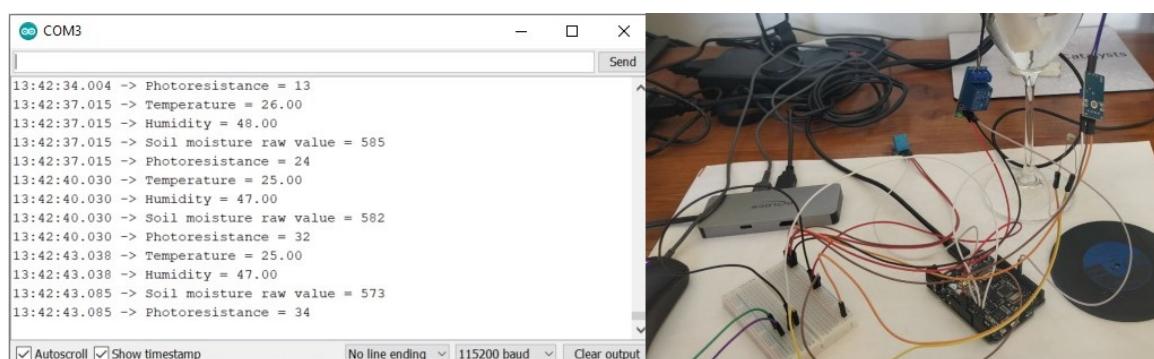


Figura 6.19 Fotorezistor și valori înregistrare în semi-întuneric

În Figura 6.20 am supus fotorezistorul la o sursă de lumină puternică. Aceasta a crescut conductivitatea fotorezistorului și putem observa cum valoarea înregistrată a crescut și ea.

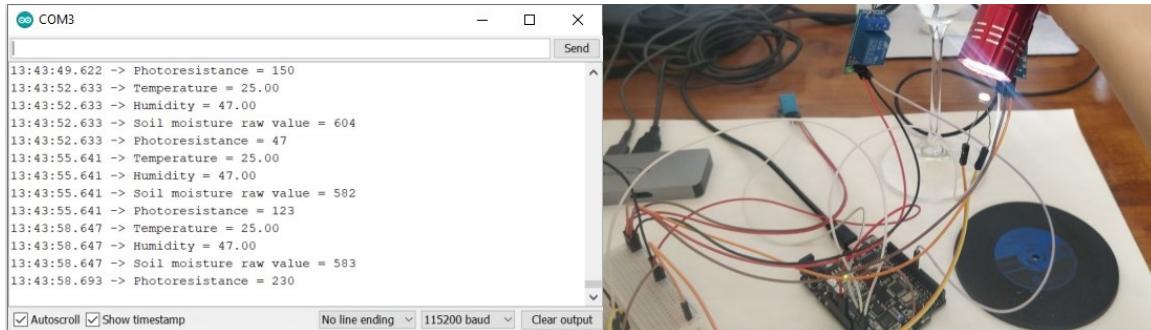


Figura 6.20 Fotorezistor și valori înregistrate în lumină puternică

Conform rezultatelor obținute în urma acestui scenariu de testare am concluzionat că fotorezistorul funcționează în mod adekvat.

### 6.3.5. Testare mini pompa de apă

Pentru testarea minipompei am apelat la un releu cu care să putem alimenta pompa de la o altă sursă de tensiune în locul plăcuței, iar plăcuța să fie responsabilă doar de controlul pompei (pornirea/oprirea acesteia). De asemenea ne-am mai folosit de modulul de umiditate a solului testat în prealabil.

Scenariul pe care l-am utilizat a fost: citirea valorii înregistrate de modulul de umiditate a solului iar când valoarea aceasta crește peste 700 (solul devine uscat, deci rezistența dintre sondele senzorului scade) activarea mini pompei submersibile.

Pentru implementarea scenariului am conectat modulul de măsurare a umidității solului la plăcuță, am conectat pompa la releu, și am legat pinul de intrare date al releeului la un pin digital, setat pentru ieșire, al plăcuței. Pompa am plasat-o într-un pahar transparent cu apă ca să observăm că întradeva pompează apa.

În Figura 6.21 putem observa că senzorul este plasat în paharul cu apă, deci înregistrează o conductivitate mare (rezistență scăzută) astfel popa este oprită.

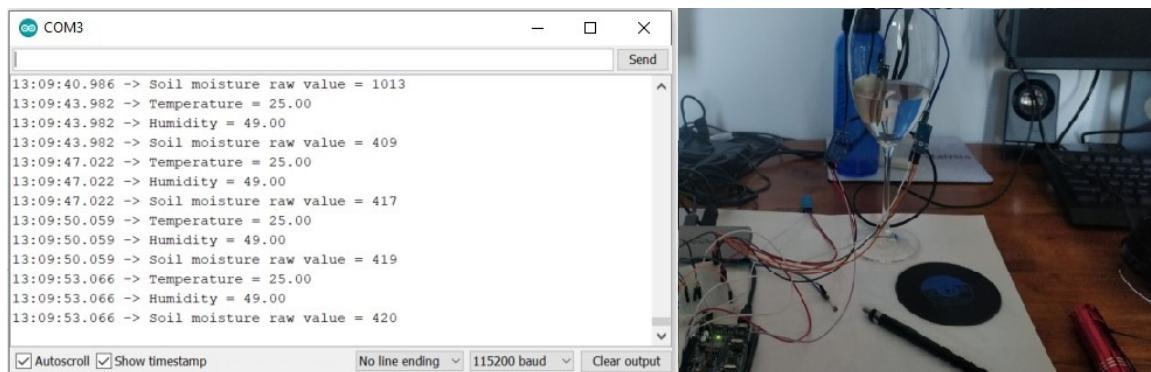


Figura 6.21 Valori înregistrate de senzori și mini pompa oprită

În Figura 6.22 observăm că am scos senzorul din pahar și l-am lăsat cu sondele în aer, unde acesta înregistrează o rezistență mult mai mare (conductivitatea aerului fiind mult mai mică decât a apei) astfel că mini pompa a început să pomeze apă în pahar.

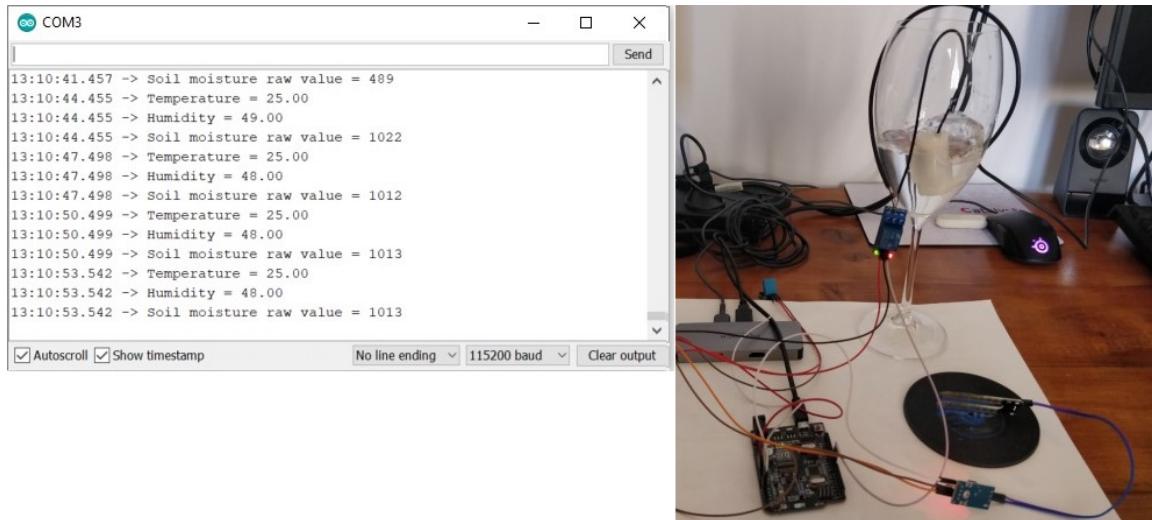


Figura 6.22 Valori înregistrate de senzori și mini pompă pornită

Conform rezultatelor obținute în urma acestui scenariu de testare am concluzionat că mini pompa de apă funcționează în mod adecvat.

## Capitolul 7. Manual de Instalare si Utilizare

### 7.1. Resurse necesare pentru instalare

Întrucât aplicația este disponibilă în cloud singurul element de care utilizatorul are nevoie pentru a acces aplicația este un dispozitiv cu acces la internet.

Pentru utilizarea modului hardware, este nevoie, evident de modulul hardware în sine care vine configurat și pre-setat. Este nevoie doar să fie conectat la o sursă de tensiune.

### 7.2. Manual de utilizare pentru utilizator neautentificat

Orice utilizator, indiferent dacă are sau nu cont în sistemul nostru poate accesa pagina de home vizibila în Figura 7.1 prin accesarea aplicației.

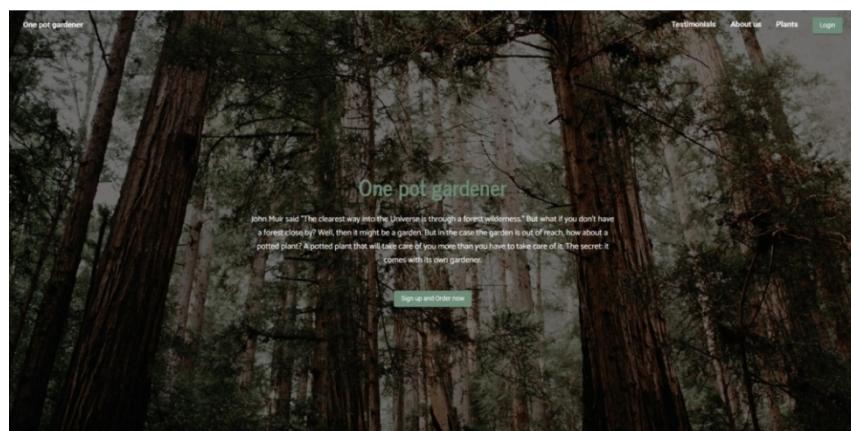


Figura 7.1 Pagina de home

O altă funcționalitate de care poate beneficia un utilizator neautentificat este vizualizarea listei de plante. Pentru aceasta el poate naviga la lista de plante după cum este evidențiat în Figura 7.2. Din meniu sau din secțiunea de prezentare.

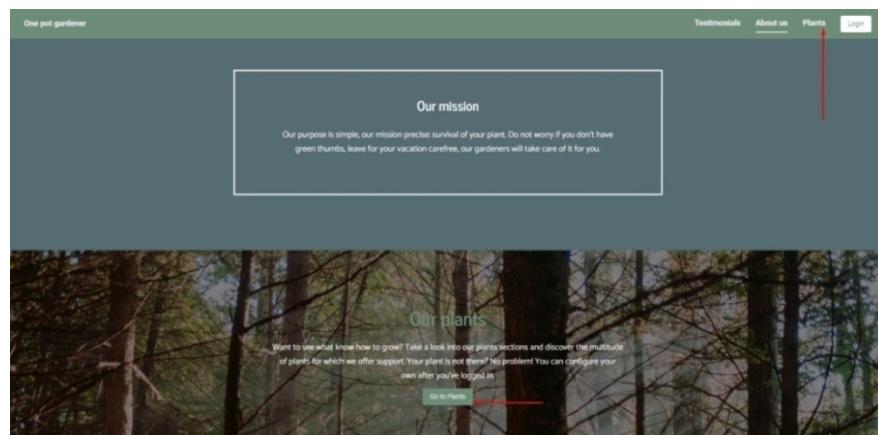


Figura 7.2 Navigare la lista de plante

Aceste acțiuni redirecționează utilizatorul la lista de plante. Aceasta este vizibilă în Figura 7.3.



Figura 7.3 Lista de tipuri de plante disponibile în aplicație

Odată ajuns în lista de plante, pentru a vedea nevoile de creștere a unui tip de plante, utilizatorul trebuie să apese pe butonul de tip dropdown „Requirements”. Acesta afișează apoi cerințele de creștere a plantei. A se vedea în Figura 7.4

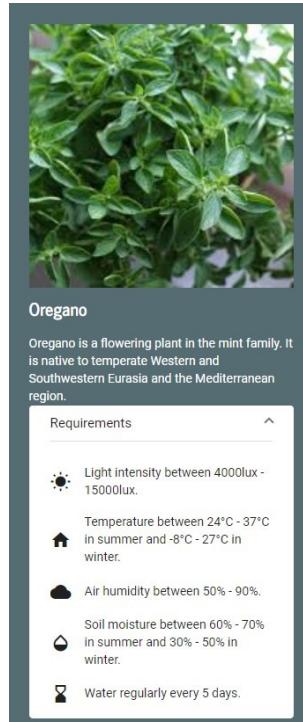


Figura 7.4 Nevoi de creștere pentru plantă

Pentru a beneficia de celălalte funcționalități ale aplicației utilizatoru trebuie să se înregistreze și autentifice în aplicație. Pentru înregistrare sau autentificare, utilizatorul trebuie să accesese pagina de login. Aceasta poate fi accesată din meniu sau din pagina de prezentare după ce este vizibil în Figura 7.5.

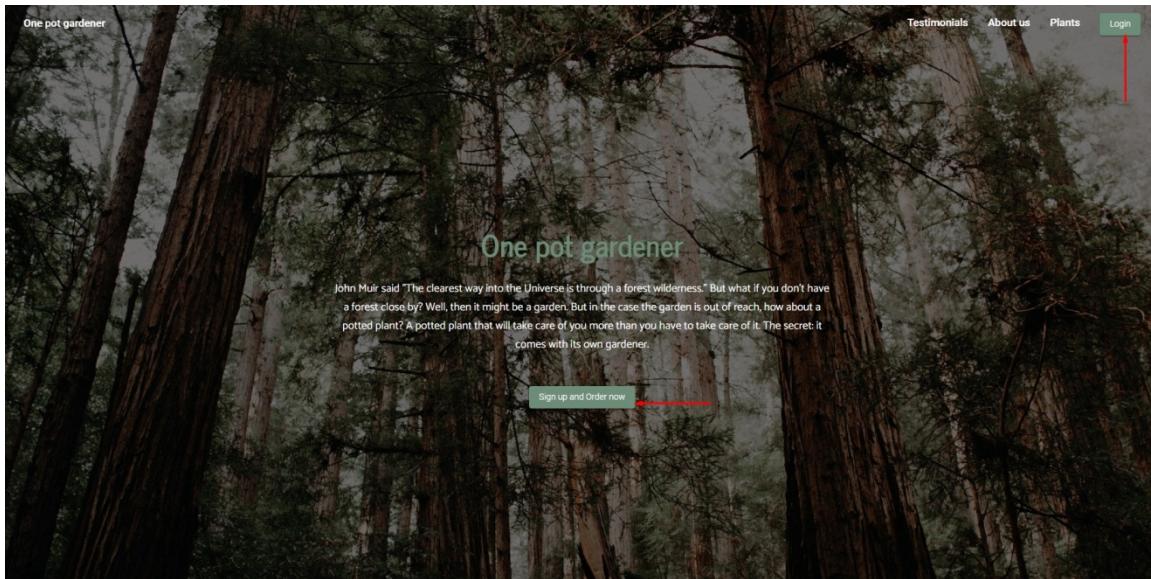


Figura 7.5 Accesare pagina de autentificare

Această acțiune va redirecționa utilizatorul la pagina de autentificare. Acesta trebuie să accesese secțiunea de înregistrare care este vizibilă în Figura 7.6 și să completeze câmpurile necesare. O dată efectuați acești pași cu succes, utilizatorul va deține un cont în aplicație.

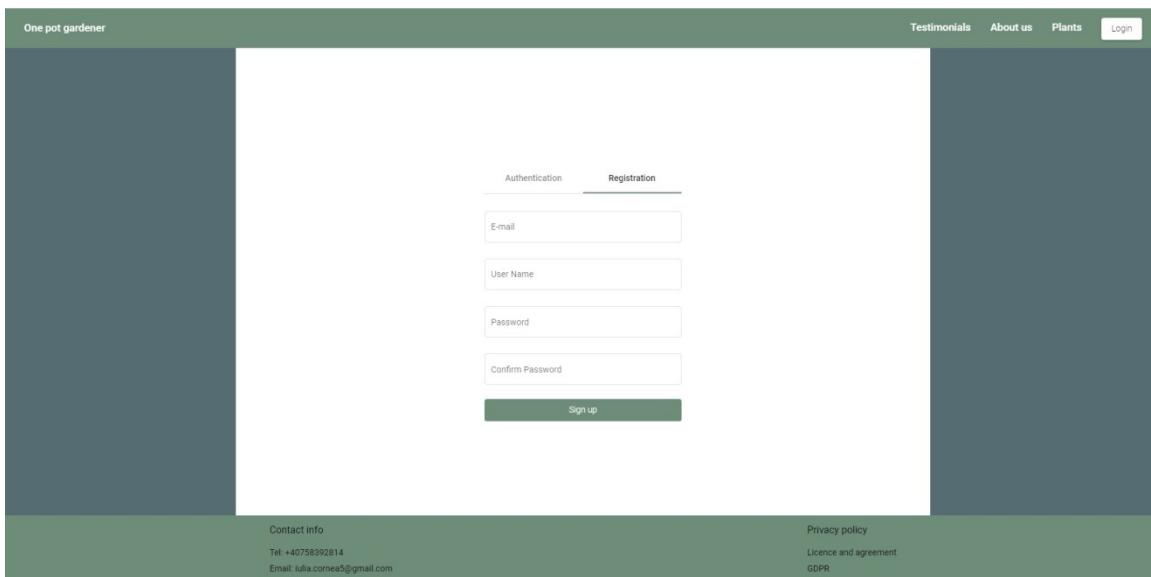


Figura 7.6 Pagină de autentificare – secțiune de înregistrare

### 7.3. Manual de utilizare pentru utilizator înregistrat

Primul pas pe care un utilizator înregistrat trebuie să îl facă pentru a beneficia de noi funcționalități este să se autentifice în pagina de login. Aceasta este accesibilă conform Figura 7.5 iar rezultatul este redirecționare spre pagina de login, secțiunea de autentificare, vizibilă în Figura 7.7.

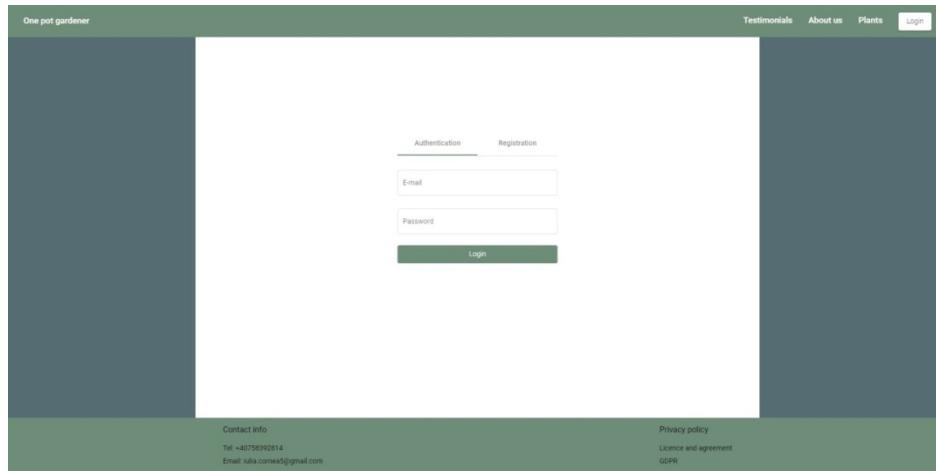


Figura 7.7 Pagina de autentificare – secțiunea de autentificare

Odată autentificat utilizatorul este redirecționat spre pagina contului său Figura 7.8, unde poate

- să vizualizeze plantele sale și specificațiile pentru acestea
- să adauge o nouă plantă
- să se delogheze
- să comande un grădinar semiautomat

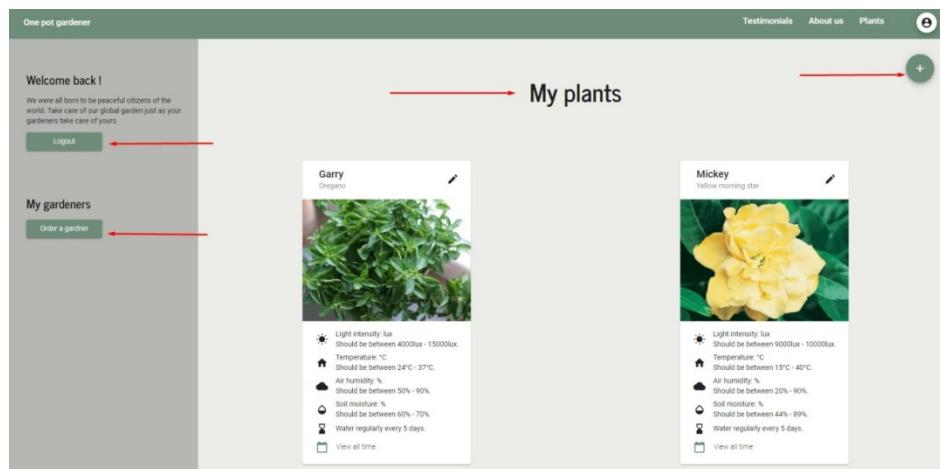


Figura 7.8 Profilul unui utilizator

Pentru adăugarea unui nou specimen în lista de plante a utilizatorului acesta accesează butonul mare cu un semnul plus pe el, care îl redirecționează la formularul de

## Capitolul 7

adăugare specimen. O dată completat formularul vizibil în Figura 7.9 utilizatorul este redirecționat spre contul său unde poate observa noua plantă.

The screenshot shows a web page titled 'Add a new home plant'. At the top, there is a navigation bar with links for 'Testimonials', 'About us', and 'Plants', along with a user icon. The main form has fields for 'Name of the plant' (containing 'Mouse'), 'Season' (set to 'Summer'), 'Plant type' (set to 'Yellow morning star'), and 'Gardener'. A red arrow points from the text 'adăugare specimen' to the 'Save' button at the bottom right. Below the form is a footer with 'Contact info' (Tel: +40758392814, Email: Iulia.cornea5@gmail.com) and 'Privacy policy' (GDPR).

Figura 7.9 Adăugare un nou specimen de plantă

În figura Figura 7.10 observă planta că salvată a fost adăugată în lista de plante a utilizatorului.

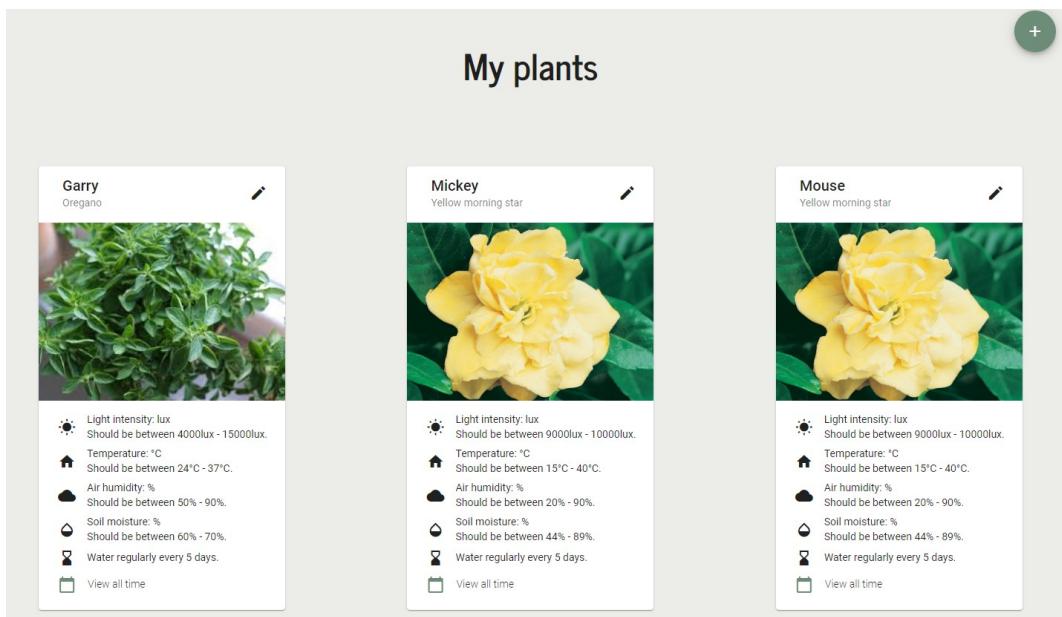


Figura 7.10 Plantă adăugată cu succes în lita utilizatorului

Pentru comandare unui grădinar semiautomat utilizatorul trebuie să acceseze butonul „Order a gardener” din contul său, Figura 7.8 care va redirecționa utilizatorul spre formularul comandă a unui grădinar semiautomat. Acest formular are doar câteva câmpuri obligatorii, Figura 7.11 deoarece în această fază a aplicației nu ne-am axat pe un sistem automat de comenzi, după ce utilizatorul a plasat o comandă va fi contactat de un

reprezentat de vânzări și va avea o conversație cu acesta pentru continuarea procesului de comandă și livrare.

Contact info  
Tel: +4075892814  
Email: iulia.cornei5@gmail.com

Privacy policy  
Licence and agreement  
GDPR

Figura 7.11 Formulare de comandare grădinar semi-automat

#### 7.4. Manual de utilizare pentru administrator

Rolul administratorului este să adauge plante noi în aplicație și să confirme comenzi de livrare. Odată autentificat, din pagina de plante, administratorul va putea adăuga plante noi prin apăsarea butonul de add, a se vedea în Figura 7.12.

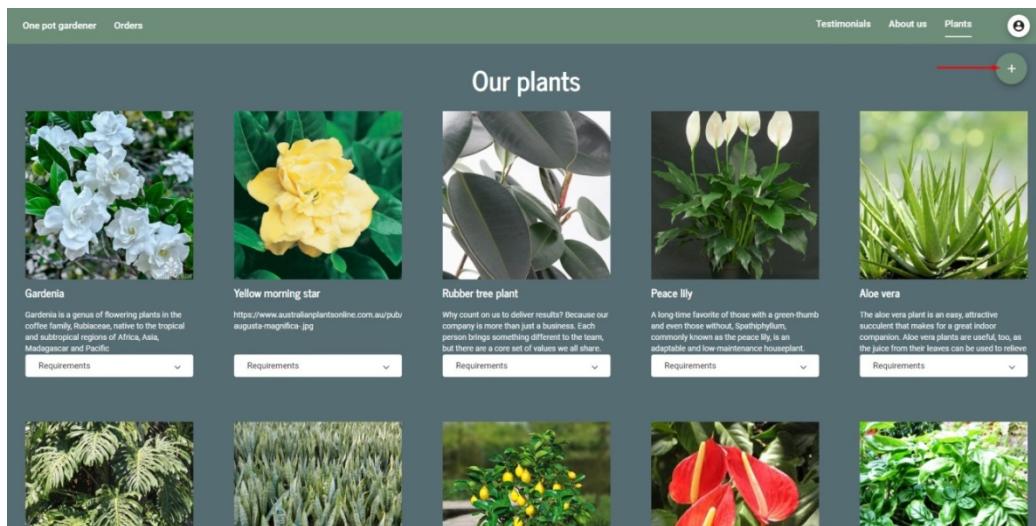


Figura 7.12 Accesare formular de adăugare plantă

Această acțiune va redirectiona utilizatorul spre formularul de creare a unei plante noi. După ce a completat secțiunea de detalii ale plantei din Figura 7.13 și secțiunea de configurații de creștere din Figura 7.14, acesta va salva planta, Figura 7.15.

## Capitolul 7

The screenshot shows a web page titled 'Add a new plant type'. At the top, there are two tabs: 'Details' and 'Growing', with 'Growing' being the active tab, indicated by a red arrow. Below the tabs, there are fields for 'Plant name' (containing 'Oregano') and 'Plant description' (containing 'Oregano is a flowering plant in the mint family. It is native to temperate Western and Southwestern Eurasia and the Mediterranean regions.'), along with a 'Link to plant image' field containing a placeholder URL. A 'Save' button is located at the bottom right of the form.

Figura 7.13 Adăugare o nouă plantă – secțiune detailii

The screenshot shows the 'Add a new plant type' form with the 'Growing' tab selected. Under the 'Growing configuration name' field, 'Mediterranean plants' is entered. The 'Watering interval in days' is set to 5. The 'Air humidity' range is from 50% to 90%. The 'Surrounding temperature in hot season' range is from 24°C to 37°C. The 'Surrounding temperature in cold season' range is from -8°C to 27°C. The 'Light intensity' range is from 4000 to 15000 LUX. The 'Soil moisture in hot season' range is from 60% to 70%. The 'Soil moisture in cold season' range is from 30% to 50%. A 'Save' button is located at the bottom right.

Figura 7.14 Adăugare o nouă plantă – secțiune configurații de creștere

The screenshot shows the 'Add a new plant type' form with the 'Growing' tab selected. The configuration settings are identical to Figura 7.14. A red arrow points to the 'Save' button at the bottom right of the form.

Figura 7.15 Salvare o nouă plantă

Pentru vizualizarea listei de comenzi utilizatorul trebuie să acceseze pagina de comenzi prin apăsare butonul „Orders” din meniu. Acesta va fi redirectat la pagina de comenzi unde poate schimba statusul acestora apăsând pe butonul corespunzător. O

confirmare imediată vine prin reîncărcarea paginii și vizualizarea noului status în coloana corespunzătoare. A se vedea Figura 7.16.

ID	Gardener name	Created at	Status	Approve	Complete	Cancel
5f05c27c-0c91-498b-a443-fb81aaef41db	Gelu	2020-09-05T22:00:19.876772+03:00	COMPLETED		Complete	Cancel
b52679e-995-44ff-ac0a-25a68d9394	2423421	2020-09-05T21:53:45.542692+03:00	PENDING_APPROVAL	Approve	Complete	Cancel
d96c98a-2250-48e7-8679-5c084ee42544	12342134	2020-09-05T21:53:41.16362+03:00	COMPLETED	Approve	Complete	Cancel
e41cf31c-7270-4977fb49-0ef648204816	1234	2020-09-05T21:53:36.980496+03:00	PENDING_APPROVAL	Approve	Complete	Cancel
988d47f-ab57-41af-a214-e095faaf5a9	Admin gardener	2020-09-05T17:52:05.400327+03:00	COMPLETED	Approve	Complete	Cancel
b0895f-5fcfa-419f-9c70-3febd77392	Prisca	2020-09-05T10:10:02.12189+03:00	COMPLETED	Approve	Complete	Cancel
8995a70e-8e63-4d65-e3f0-eccaff2d554a	Elvis	2020-09-05T15:53:25.804268+03:00	COMPLETED	Approve	Complete	Cancel
b161d8b-98e7-4716-aec6-09ce39a329e6	Santa Maria de la Cruz	2020-09-05T15:49:38.089931+03:00	CANCELLED	Approve	Complete	Cancel
0ff77eb-d424-4cb1-9112-f45e59671621	Jose Mandinga	2020-09-05T09:36:20.284384+03:00	COMPLETED	Approve	Complete	Cancel

Figura 7.16 Listă de comenzi

Odată ce o comandă este completată (butonul „COMPLETE”) utilizatorul care a comandat grădinarul va putea vedea noul să grădinar în secțiunea „My gardeners” din pagina contului său.

## 7.5. Manual de utilizare pentru utilizator hardware

Un utilizator care are o comandă completă a unui grădinar semiautomat poate să îl vadă pe acesta în secțiunea „My gardeners” după cum se observă în figura. Figura 7.17

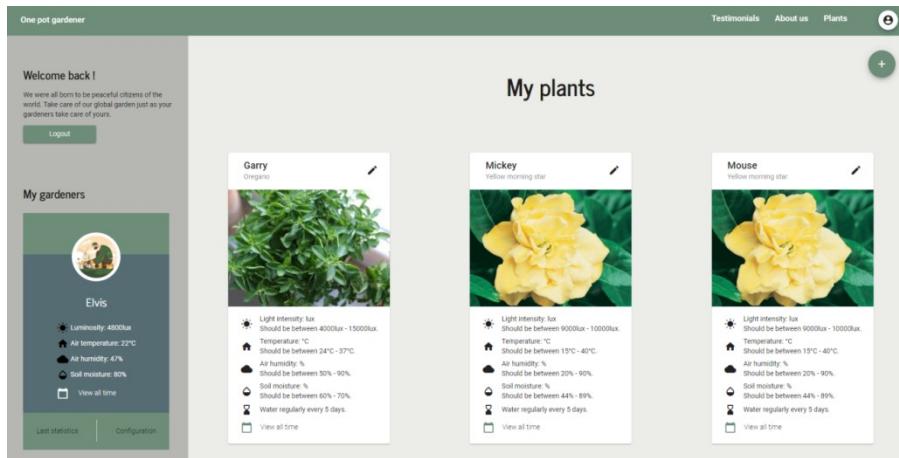


Figura 7.17 Pagină utilizator hardware

Despre un grădinar, utilizatorul poate primi mai multe informații după cum vedem în Figura 7.18. În cazul în care un grădinar nu a trimis încă nicio statistică, acesta arată ca

în stânga. Apăsând butonul de „Configuration: putem vedea valorile cu care am configurat grădinarul ca acesta să se poată conecta la rețeaua de internet de acasă (centru) iar o dată ce am conectat grădinarul la curent, acesta emite informații care sunt vizibile dacă apăsăm pe butonul „Last statistics”(dreapta).



Figura 7.18 Grădinar semiautomat – stări

Pentru utilizarea sistemului hardware utilizatorul va trebui să conecteze atât plăcuța cât și pompa la o sursă de curent și să se asigure că plăcuța e setată în modul descris în Figura 7.19.

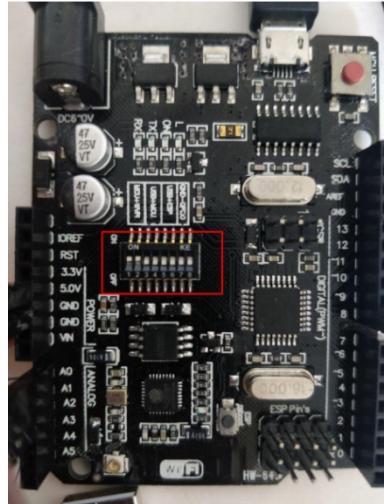


Figura 7.19 Setare plăcuță

Mai apoi, utilizatorul trebuie să introducă senzorul de umiditate a solului adânc în solul plantei cu grijă a nu rupe sondele acestuia. A se vedea Figura 7.20. Apoi, să scufunde pompa de apă într-un recipient cu apă și sa pună furtunul cu capetele perforate pe suprafața ghiveciului.



Figura 7.20 Senzor de umiditate a solului introdus corespunzător

Acum că grădinarul semiautomat trimite statistici relevante ele pot fi vizualizate în pagina de statistici, apăsând butonul „View all time” de pe cardul unui grădinar. Vezi Figura 7.21.

One pot gardener		One pot statistics from Jose Mandinga the gardener					
Received at	Soil moisture	Light intensity	Air temperature	Air humidity	Soil moisture (raw value)	Light intensity (raw value)	
05 Sep 14:05	100%	50000lux	22°C	47%	1018	636	
05 Sep 14:04	0%	50000lux	22°C	47%	530	471	
05 Sep 14:04	0%	50000lux	22°C	47%	529	471	
05 Sep 14:04	0%	50000lux	22°C	47%	529	471	
05 Sep 14:04	0%	50000lux	22°C	47%	529	471	
05 Sep 14:04	0%	50000lux	22°C	48%	529	472	
05 Sep 14:03	0%	50000lux	22°C	48%	529	471	
05 Sep 14:03	0%	50000lux	22°C	48%	529	471	
05 Sep 14:03	0%	50000lux	22°C	48%	529	471	
05 Sep 14:03	0%	50000lux	22°C	48%	529	471	
05 Sep 14:03	0%	50000lux	22°C	47%	529	471	
05 Sep 14:02	0%	50000lux	22°C	47%	529	471	
05 Sep 14:02	0%	50000lux	22°C	47%	529	471	

Figura 7.21 Statistici oferite de un grădinar semiautomat

## Capitolul 8. Concluzii

### 8.1. Contribuții personale

Ideea de a implementa un sistem care să poată oferi susținere în creșterea plantelor de apartament a venit în special din necesitatea de a păstra plantele în viață în perioada concediilor sau a deplasărilor de acasă. Am luat în considerare diversitatea de plante din apartament și am realizat că fiecare are nevoi diferite din punct de vedere a frecvenței de udare, a solului sau a luminii. Astfel s-a născut conceputl Grădinarului de ghiveci.

Contribuțiiile personale în lucrarea de față constă în alegerea temei, proiectarea arhitecturii sistemului, alegerea resurselor pentru implementare și implementarea propriu-zisă a arhitecturii respective. Pentru proiectarea arhitecturii sistemului am studiat și folosit arhitecturi consacrate precum arhitectura client – server, arhitectura pe niveluri, etc. În alegerea resurselor de dezvoltare s-au studiat mult ultimele tendințe din industrie, întrucât resursele cele mai bune ajung să câștige popularitate din ce în ce mai mare ne-am putut baze pe faptul că o resursă utilizată la scară largă va fi ușor de folosit și va avea mai mult suport în situațiile de depanare.

### 8.2. Analiza rezultatelor obținute

După cum am menționat în Capitolul 2, obiectivele acestui proiect sunt:

- Monitorizarea mediului de dezvoltare a plantei
- Controlul mediului
- Stocarea și deservirea de informații despre plante și la mediul acestora
- Oferirea unei interfețe web prin care utilizatorul poate comunica cu sistemul

Monitorizarea sistemului se face în timp real și măsoară patru parametri despre sistem:

- Temperatura ambientală
- Nivelul de umiditate aerului
- Intensitatea luminoasă
- Nivelul de umiditate a solului

Având în vedere că aceștia sunt parametri esențiali pentru dezvoltarea propice a unei plante de apartament considerăm că proiectul a atins sопul de monitorizare a mediului de dezvoltare a plantei.

Controlul mediului implică controlul doar a uneia dintre cei patru parametri: umiditatea solului. Deși doar unul dintre parametri poate fi controlat, acesta este cel mai ușor de neglijat de către utilizator și cel mai greu de întreținut. Datorită acestui aspect, considerăm că sistemul oferă suficient control asupra mediului plantei.

Pentru obiectivul de stocare și deservirea de informații despre plante și mediul propice de dezvoltare a acestora s-a stocat un număr limitat de plante în sistemul de față. Din acest punct de vedere considerăm obiectivul parțial atins, sistemul oferind o bază de date incompletă utilizatorilor. În remedierea acestui neasuns oferim utilizatorilor

posibilitatea de a adăuga ei însăși tipuri noi de plante și informații despre creșterea acestora.

Pentru partea de interacțiune a utilizatorului cu sistemului considerăm că am atins așteptările initiale, rezultatul final fiind prezentat într-o manieră prietenoasă utilizatorului, ușor de folosit și atractiv din punct de vedere vizual.

### **8.3. Dezvoltări ulterioare**

Având în vedere că sistemul proiectat este un prototip, acesta poate beneficia de îmbunătățiri atât pe partea de software cât și pe partea de hardware. De asemenea, aceste îmbunătățiri vizează atât cerințele funcționale cât și cele non-funcționale.

#### *8.3.1. Dezvoltări software*

Pentru partea software evidențiem următoarele dezvoltări ale aplicației din punct de vedere a funcționalității:

- Funcția de identificare a unei plante – această funcție ar oferi utilizatorilor un mod rapid de a identifica o plantă după cum este prezentat și în [9] (și implicit nevoile acesteia). Astfel că aceștia pot alege în cunoștință de cauză dacă se pretează achiziționarea plantei sau nu apartamentului lor.
- Funcția de cont partajat sau de partajare a plantelor pentru utilizatorii care locuiesc în același apartament astfel încât aceștia să poată vizualiza date despre plantele comune
- Implementarea notificărilor în timp real prin notificări în browser
- Funcția de diagnosticare a plantei prin procesarea de imagini – în unele cazuri neajunsurile plantelor se datorează altor factori decât cei monitorizați, cum ar fi tipului de sol în care sunt plantate. Această funcționalitate ar ajuta utilizatorul în identificarea și remedierea unei astfel de probleme.

#### *8.3.2. Dezvoltări hardware*

Pentru modulul hardware cea mai importantă dezvoltare pe care o putem face este reducerea costului de achiziție. Aceasta înseamnă de exemplu că am putea folosi module independente, în locul unei plăcuțe R3 UNO + WiFi, pe care le-am conectat manual între ele.

O altă îmbunătățire ar putea fi reducerea consumului de energie electrică care ar putea fi implementat prin limitarea numărului de comunicări pe care îl are modulul ESP8266 cu serverul sau prin utilizarea altui protocol de comunicare care nu trimite mesaje la fel de mari ca HTTP. Un exemplu ar fi MQTT.

O îmbunătățire funcțională dar care implică și parte de hardware ar putea fi controlul și programarea sistemului să irige cu o anumită cantitate de apă la un anumit interval de timp.

Desigur, sistemul poate fi dezvoltat și prin extinderea numărului de parametri pe care acesta îi monitorizează. Un parametru relevant ar fi aciditatea solului. Dacă, în schimb adăugăm o cameră de luat vederi, combinat cu un algoritm de procesare de

imaginii am putea supraveghea planta automat și identifica dacă aceasta crește suficient de repede, nu are pete de arsuri solare, etc. Pe aceeași parte de extindere hardware sistemul ar putea beneficia de extinderea componentelor pe care le poate controla asupra mediului, aceasta ar putea transforma Grădinarul de ghiveci chiar într-o mini seră de apartament.

## Bibliografie

- [1] Yuval N. Harari, „*Sapiens : a Brief History of Humankind*”, New York :Harper, 2015.  
ISBN-13 : 978-0062316097
- [2] Mattern, F., Flörkemeier, C. „Vom Internet der Computer zum Internet der Dinge”, in *Informatik Spektrum* 33, pp. 107–121, 2010  
DOI: 10.1007/s00287-010-0417-7  
Available at: <http://www.vs.inf.ethz.ch/publ/papers/Internet-of-things.pdf>
- [3] Rad, C.R., Hancu, O., Takacs, I.A., Olteanu, G.: Smart monitoring of potato crop: a cyber-physical system architecture model in the field of precision agriculture. *A. A. Sci. Proc.* 6, pp 73–79, 2005  
DOI: 10.1016/j.aaspro.2015.08.041  
Available at: <https://www.sciencedirect.com/science/article/pii/S2210784315001746>
- [4] Raj, Jennifer & Ananthi, Vijitha. „Automation using IOT in greenhouse environment”, in *Journal of Information Technology and Digital World*. 01. pp. 38-47.  
DOI: 10.36548/jitdw.2019.1.005.  
Available at: [https://www.researchgate.net/publication/337715165\\_AUTOMATION\\_USING\\_IOT\\_IN\\_GREENHOUSE\\_ENVIRONMENT](https://www.researchgate.net/publication/337715165_AUTOMATION_USING_IOT_IN_GREENHOUSE_ENVIRONMENT)
- [5] S Aishwarya, R. Thiuzhika, A. Manepally, „IoT based greenhouse”, in *International Journal of Advance Research, Ideas and Innovations in Technology, Volume 4, Issue 5*, pp 374-375, 2018.  
Available at: <https://www.ijariit.com/manuscripts/v4i5/V4I5-1300.pdf>
- [6] N. Thirer, I. Uchansky, „An FPGA Based Computer System for Greenhouse Control”, in *Athens Journal of Sciences, Volume 2, Issue 1*, pp. 23-32, 2015.  
DOI: 10.30958/ajs.2-1-3  
Available at: <https://www.athensjournals.gr/sciences/2015-2-1-3-Thirer.pdf>
- [7] Shreyas Bhujbal, Yash Deshpande, Arpit Gupta, Ojas Bhelsekar, „IOT Based Smart Greenhouse”, in *International Journal of Innovative Research in Science, Engineering and Technology, Vol. 7, Issue 1*, January 2018  
Available at: [http://www.ijirset.com/upload/2018/january/74\\_IOT.pdf](http://www.ijirset.com/upload/2018/january/74_IOT.pdf)
- [8] Robert C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship. Upper Saddle River, NJ: Prentice Hall, 2009.  
ISBN-13: 978-0132350884

- [9] N.Valliammal, S.N.Geethalakshmi, „Automatic Recognition System Using Preferential Image Segmentation For Leaf And Flower Images”, in *Computer Science & Engineering: An International Journal (CSEIJ)*, Vol.1, No.4, pp. 13 – 25, 2011  
Available at:  
[https://www.researchgate.net/publication/267944144\\_Automatic\\_Recognition\\_System\\_Using\\_Preferential\\_Image\\_Segmentation\\_For\\_Leaf\\_And\\_Flower\\_Images](https://www.researchgate.net/publication/267944144_Automatic_Recognition_System_Using_Preferential_Image_Segmentation_For_Leaf_And_Flower_Images)

**Anexă 1 – Lista figurilor**

Figura 1.1 Componete ale unui sistem IOT.....	2
Figura 1.2 Seră inteligentă .....	3
Figura 1.3 Sistem IOT cu ceainic intelligent .....	4
Figura 2.1 Sistem de grădinărit inteligent.....	5
Figura 3.1 Raspberry Pi 3 .....	10
Figura 3.2 Arduino UNO .....	11
Figura 3.3 Netduino 3 WiFi.....	11
Figura 3.4 Plăcuță FPGA .....	12
Figura 3.5 UNO + Wi-Fi R3 .....	12
Figura 3.6 Garden Answers Plant Id – selectarea plantei .....	14
Figura 3.7 Vera – profilul unei plante.....	15
Figura 4.1 Arhitectura conceptuală a sistemului.....	16
Figura 4.2 Fluxul de business al părții software a aplicației.....	17
Figura 4.3 Fluxul de lucru al părții hardware a aplicației .....	17
Figura 4.4 Cazuri de utilizare a aplicație .....	18
Figura 4.5 Diagramă de flux pentru crearea unui specimen nou (include toate cazurile de utilizare care sunt postcondiții pentru acesta) .....	25
Figura 4.6 Proces de achiziție a unui Grădinar de ghiveci și vizualizarea datelor furnizate de acesta.....	27
Figura 4.7 Modulul de grădinărit semi-automat .....	30
Figura 5.1 Arhitectură conceptuală a Grădinarului de ghiveci .....	32
Figura 5.2 Arhitectura detaliată a sistemului .....	33
Figura 5.3 Arhitectura aplicației server .....	34
Figura 5.4 Diagramă de secvență pentru cazul de utilizare „Utilizatorul salvează un nou specimen”.....	35
Figura 5.5 Generarea scheletului aplicației server.....	37
Figura 5.6 OpenAPI Specification pentru aplicația server interpretată de Editorul Swagger.....	38
Figura 5.7 Task-uri Gradle puse la dispoziție de OpenAPI Tools .....	38
Figura 5.8 Configurarea task-ului openApiGenerate pentru generarea codului de API .....	39
Figura 5.9 Specificație OpenAPI (stânga) și cod kotlin-spring generat (dreapta) ..	39
Figura 5.10 Beneficiile Spring Data JPA evidențiate prin claritatea codului .....	40
Figura 5.11 Diagrama relațională a entităților din baza de date .....	41
Figura 5.12 Configurarea Flyway .....	41
Figura 5.13 Migrări executate pe baza de date a sistemului .....	42
Figura 5.14 Arhitectura MVC în Angular .....	43
Figura 5.15 Componentele aplicației client .....	44
Figura 5.16 Specificație OpenAPI (stânga) și codul generat de generatorul typescript-angular (dreapta) .....	45
Figura 5.17 Modificarea fișierului .npmrc pentru utilizarea registry-ului local oferit de Verdaccio.....	45
Figura 5.18 Comunicarea între modulul hardware și aplicația sever.....	46
Figura 5.19 Moduri funcționare plăcuță R3 Uno + WiFi .....	46

Figura 5.20 Modul hardware – Responsabilități.....	47
Figura 5.21 Senzor DHT11 .....	48
Figura 5.22 Modul cu senzor de umiditate sol FC28 și cip LM393 .....	48
Figura 5.23 Fotrezistor.....	49
Figura 5.24 Mini pompa submersibilă și Modul releu de 5V .....	49
Figura 5.25 Comunicare între două componente UART .....	51
Figura 6.1 Trimitere apel http GET către server din Postman.....	54
Figura 6.2 Trimitere apel http POST către server din Postman.....	54
Figura 6.3 Baza de date vizibilă în aplicația pgAdmin.....	55
Figura 6.4 Configurări bază de date h2.....	55
Figura 6.5 Scenarii de test pentru componenta User .....	56
Figura 6.6 Test pentru delogarea unui utilizator.....	56
Figura 6.7 Setările pentru utilizare ATmega328P .....	57
Figura 6.8 Testare ATmega328P .....	57
Figura 6.9 Setările pentru utilizare ESP2866.....	58
Figura 6.10 Testare ESP8266 .....	58
Figura 6.11 Serve mock Postman și Monitorul serial Arduino .....	59
Figura 6.12 Plăcuță setată în modul de operare „Mega328+ESP8266” .....	60
Figura 6.13 Rezultatele testării comunicării dintre modulul Atmeg328P și ESP8266.....	60
Figura 6.14 Măsurători înregistrare de senzorul DHT11 sub acțiunea umezirii...	61
Figura 6.15 Măsurători înregistrarea de senzorul DHT11 sub acțiunea încălzirii	61
Figura 6.16 Măsurători înregistrate de modulul de umiditate a solului într-un mediu uscat .....	62
Figura 6.17 Măsurători înregistrare de modulul de umiditate a solului într-un mediu umed.....	62
Figura 6.18 Schemă petnru conectarea fotrezistorului și rezistenței .....	63
Figura 6.19 Fotrezistor și valori înregistrare în semi-întuneric .....	63
Figura 6.20 Fotrezistor și valori înregistrate în lumină puternică.....	64
Figura 6.21 Valori înregistrate de senzori și mini pompa oprită .....	64
Figura 6.22 Valori înregistrate de senzori și mini pompă pornită .....	65
Figura 7.1 Pagina de home.....	66
Figura 7.2 Navigare la lista de plante .....	66
Figura 7.3 Lista de tipuri de plante disponibile în aplicație.....	67
Figura 7.4 Nevoi de creștere pentru plantă .....	67
Figura 7.5 Accesare pagina de autentificare .....	68
Figura 7.6 Pagină de autentificare – secțiune de înregistrare .....	68
Figura 7.7 Pagina de autentificare – secțiunea de autentificare.....	69
Figura 7.8 Profilul unui utilizator .....	69
Figura 7.9 Adăugare un nou specimen de plantă.....	70
Figura 7.10 Plantă adăugață cu succes în lita utilizatorului.....	70
Figura 7.11 Formulare de comandare grădinar semi-automat .....	71
Figura 7.12 Accesare formular de adăugare plantă .....	71
Figura 7.13 Adăugare o nouă plantă – secțiune detailii.....	72
Figura 7.14 Adăugare o nouă plantă – secțiune configurații de creștere.....	72
Figura 7.15 Salvare o nouă plantă.....	72

Figura 7.16 Listă de comenzi.....	73
Figura 7.17 Pagină utilizator hardware.....	73
Figura 7.18 Grădinar semiautomat – stări.....	74
Figura 7.19 Setare plăcuță.....	74
Figura 7.20 Senzor de umiditate a solului introdus corespunzător .....	75
Figura 7.21 Statistici oferite de un grădinar semiautomat .....	75

## Anexă 2 – Lista tabelelor

Tabel 3.1 Comparare între dispozitivele hardware de control .....	13
Tabel 4.1 Caz de utilizare – Înregistrarea unui utilizator în aplicație .....	19
Tabel 4.2 Caz de utilizare – Autentificarea utilizatorilor în aplicație .....	20
Tabel 4.3 Caz de utilizare – Vizualizare listă globală de plante .....	21
Tabel 4.4 Caz de utilizare – Adăugare plantă în lista globală .....	21
Tabel 4.5 Caz de utilizare – Vizualizare grădina virtuală .....	22
Tabel 4.6 Caz de utilizare – Adăugare plantă în grădina virtuală .....	22
Tabel 4.7 Caz de utilizare – Creare plantă configurată manual .....	23
Tabel 4.8 Caz de utilizare – Vizualizare informații plantă din grădină .....	24
Tabel 4.9 Caz de utilizare – 4.1.9. Proces de achiziție a grădinarului de ghiveci (modulul hardware) .....	26
Tabel 4.10 Caz de utilizare – Vizualizare date despre mediul planetei .....	26

### Anexă 3 – Glosar de termeni

Termen	Scrută descriere
Grădinarul de ghiveci	Denumirea sistemului în limba română
One pot gardener	Denumirea sistemului în limba engleză
IOT	Domeniu ce se ocupă cu dispozitive conectate la internet ce colectează și furnizează date sau primesc și execută acțiuni prin intermediul internetului
Modulul software	Componenta software a sistemului dezvoltat. Include atât aplicația client cât și aplicație server și funcționalitățile oferite de acestea
Modulul hardware	Componenta hardware a sistemului dezvoltat. Include atât placuța R3 UNO + WiFi cât și senzorii utilizati, pomă de apă și releul
Grădinar semiautomat	Modulul hardware al aplicației
Plantă	Un tip generic de plantă, de exemplu Aloe vera
Tip de plantă	Un tip generic de plantă, de exemplu Aloe vera
Plantă de apartament	O plantă anume, de ex planta de Aloe vera de lângă televizoul utilizatorului
Specimen	O plantă anume, de ex planta de Aloe vera de lângă televizoul utilizatorului
API	Application Programming Interface
JSON	JavaScript Object Notation
YAML	Yaml Ain't Markup Language
HTTP	Hypertext Transfer Protocol
REST	Representational State Transfer
DTO	Data Transfer Object
GET	Request HTTP care îndeplinește standardul REST, și anume accesează o resursă
POST	Request HTTP care îndeplinește standardul REST, și anume salvează o resursă
OpenAPI Specification	Specificație pentru descrierea serviciilor web de tipul REST. Original sub numele de Swagger Specification
OpenAPI Generator	Instrument care generează documente sau cod în funcție de un fișier care se supune OpenAPI Specification